蚁群智能优化方法及其应用

Ant Colony Intelligence Optimization Method and Its Applications

蚁群智能优化方法及其应用

柯良军　著

清华大学出版社
北　京

内 容 简 介

本书在简要阐述智能优化方法相关理论的基础上，介绍了蚁群智能优化方法的基本原理、算法基本要素等基本内容。同时，介绍了蚁群智能优化方法在旅行商问题、背包问题、定向问题、属性约简、卫星资源调度问题、旅游路线规划问题以及多目标组合优化问题等复杂组合优化问题的应用示例，详细阐述了蚁群智能优化方法在具体应用中的基本设计方法以及算法性能改善的有效途径。

本书适合作为从事智能优化方法及其应用研究的相关科技工作者、专业技术人员的参考书，也可作为计算机学科、控制科学等专业研究生和高年级本科生学习蚁群智能优化方法的参考书。

图书在版编目(CIP)数据

蚁群智能优化方法及其应用/柯良军著. —北京：清华大学出版社，2017
ISBN 978-7-302-46573-7

Ⅰ. ①蚁…　Ⅱ. ①柯…　Ⅲ. ①最优化算法－研究　Ⅳ. ①O242.23

中国版本图书馆 CIP 数据核字(2017)第 030241 号

责任编辑：王　芳
封面设计：常雪影
责任校对：李建庄
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：北京泽宇印刷有限公司
经　　销：全国新华书店
开　　本：155mm×235mm　　**印　张**：11.5　　**字　　数**：212 千字
版　　次：2017 年 6 月第 1 版　　**印　　次**：2017 年 6 月第 1 次印刷
印　　数：1～1000
定　　价：59.00 元

产品编号：071705-01

项 目 资 助

国家自然科学基金项目(61573277)
陕西省自然科学基金项目(2015JM6316)



最优化是人类决策的基本准则。智能优化方法作为一类重要的优化方法，通过模拟自然界中的智能行为或现象，在可接受的时间内，得到问题的满意解。智能计算具有很强的适应性，易于实现，广泛应用于工业生产和社会生活中的复杂大规模优化问题，受到国内外学术界和工业界的极大关注。

蚁群智能优化方法是一类重要的智能优化方法，已经用于解决许多复杂的优化问题。本书在总结主流智能优化方法的基础上，介绍了蚁群智能优化方法的基本思想和基本要素，同时，详细阐述了蚁群智能优化方法的算法改进和理论研究等方面的研究成果。

蚁群智能优化方法原理较简单，但实现起来却并不简单。它的成功应用依赖于使用者对算法原理、待解决问题的理解程度，也依赖于算法编程实现。本书着重讲述了作者在用蚁群智能优化方法解决旅行商问题、背包问题、定向问题、属性约简、卫星资源调度问题以及多目标组合优化问题等复杂组合优化问题时的设计思路，有助于读者更好地理解和掌握蚁群智能优化方法，并用于解决其他难题。

本书共10章。第1章讲述智能优化方法的基本概念及其重要性；第2章给出蚁群智能优化方法的基本原理和算法要素，概述其国内外研究现状；在后续的各个章节中，针对8个问题讲述如何利用蚁群智能优化方法进行算法设计和分析。

本书适合计算机、自动化等专业本科生和研究生用于了解和学习蚁群智能优化方法等智能计算方法，也可作为科研工作者和工程技术人员的参

考书。

本书得到国家自然科学基金项目(编号:61573277)和陕西省自然科学基金(编号:2015JM6316)的资助以及宇航动力学国家重点实验室开放基金的支持,在此表示诚挚感谢。本书的完成得益于冯祖仁教授、张青富教授和李晶研究员的指导。

由于作者水平有限,书中难免有各种不足,敬请读者不吝批评指正。

作　者

2017 年 3 月

目录

CONTENTS



绪　　章

1.1　引言

随着科学技术的进步,人类生存空间不断扩大,认识世界和改造世界的范围不断拓宽,各领域不断涌现大量的大规模、非线性、强约束的复杂优化问题。问题的计算复杂性给优化方法带来新的挑战。虽然经典优化方法的理论相对完备,然而这些方法一般只适用于具有特殊结构的优化问题。例如,单纯形算法仅适用于线性规划,不能用于非线性问题;梯度下降法假设待求解的问题是连续可微的[1]。在第二次世界大战中,美国数学家冯·诺伊曼(Von Neumann)等提出了蒙特卡罗(Monte Carlo)方法[2],并利用刚刚诞生的电子计算机进行模拟计算,有效地解决了原子弹研制过程中出现的大量问题。人们认识到利用计算机进行"采样"可以求解优化问题。蒙特卡罗方法启发人们重新认识优化,并致力于探索新型优化方法。

自20世纪80年代以来,尤其是在最近20年,一些与经典的数学规划截然不同的、试图通过模拟自然界中的自适应优化现象求解复杂优化问题的新型智能优化算法相继出现,如遗传算法、人工免疫算法、模拟退火算法、人工神经网络、蚁群算法、粒子群优化算法等[3,4],这些智能优化算法大大丰富了优化技术,也为那些传统最优化技术难以处理的优化问题提供了新的极具竞争力的解决方案。

在众多智能优化算法中，蚁群算法是最成功的算法之一[5]。该算法由意大利学者 Dorigo 等于 20 世纪 90 年代初提出，蚁群算法模拟蚂蚁觅食行为，将习得的信息保存在信息素场中，在产生解时，它利用问题相关的启发信息和习得的信息，而在信息加工时利用历史解信息更新信息素场，信息素场又反馈指导蚁群产生更好的解。

蚁群算法首先用于求解著名的旅行商问题（Traveling Salesman Problem，TSP），并获得了很好的计算效果[6]。此后该算法逐渐引起了国内外学者的关注，他们对该算法做了许多改进并且将其应用于更为广泛的领域，取得了一系列令人鼓舞的成果[5]。1999 年，Dorigo 等将其进一步发展成一种通用的优化技术——蚁群算法（Ant Colony Optimization，ACO），并将所有符合 ACO 框架的算法统称为蚁群算法[7]，从而为 ACO 的理论研究和算法设计提供了统一的框架。目前，蚁群算法的应用范围涉及城市给水排水问题[8]、卫星调度[9]、机器人领域[10~14]、电力系统[15~18]、故障诊断[19]、系统辨识[20]、数据挖掘[21~25]、图像处理[26~28]、航迹规划[29,30]、空战决策[31]、岩土工程[32]、化学工业[33]、生命科学[34]、布局优化[35]、控制参数优化[36,37]等科技和工程领域[38,39]。

1.2 复杂性理论的基础知识

一般地，优化问题可表述如下：

$$\min_{x\in\Omega} f(x) \tag{1-1}$$

其中，f 是目标函数，x 是变量，Ω 是解空间，它由一组约束定义。称解 x^* 为全局最优的（globally optimal），如果解 $x^*\in\Omega$ 比解空间中任意解的目标函数值小，即 $f(x^*)\leqslant f(x)$，$\forall x\in\Omega$；称解 x^* 为子空间 $\Theta\in\Omega$ 中局部最优的（locally optimal），如果解 $x^*\in\Theta$ 比子解空间 Θ 中任意解的目标函数值小，即 $f(x^*)\leqslant f(x)$，$\forall x\in\Theta$。

在优化问题中，目标函数、约束函数和变量有不同的表现形式，对应于不同类型的优化问题。

- 函数类型：有许多分类标准。例如按照是否线性分为线性函数或非线性函数，相应的优化问题分别称为线性优化和非线性优化；按照是否为凸分为凸函数或非凸函数，相应的优化问题分别称为凸优化和非凸优化；按照是否连续可以分为连续函数和非连续函数，相应的优化问题分别称为连续优化和非连续优化。
- 函数表示：在一些问题中，函数可以给出解析表达式。但是在一些

问题中,函数只能用黑盒子表示,通常只有若干采样点对应的目标函数值,例如结构设计问题。

- 目标函数个数:如果目标函数只有一个,则是单目标优化问题;如果不止一个,则为多目标优化问题。
- 函数系数类型:系数可能是确定的,也可能是动态变化的,还可能是不确定的(随机数、模糊数)。
- 变量类型:如果变量是函数,则是变分问题;如果变量是数值型,则又可以分为连续变量和离散变量;如果一个问题中既有连续变量也有离散变量,则该问题是混合型的。
- 变量个数:如果只有一个变量,则称为单变量问题,否则是多变量问题。

我们不禁要问:这些优化问题哪些是容易的,哪些是复杂的?下面依据计算复杂性理论[40]探讨这个问题。

计算复杂性理论研究至少需要多少的资源计算一类问题。所谓资源,通常是指时间和空间,即求解问题时所需的运算数和内存。相应地,复杂性分析包括时间复杂性和空间复杂性分析。

一个问题的复杂度不是指特定算法求解某个算例所需的资源,而是指求解该问题最优算法的复杂度。下面简单介绍算法复杂度和问题复杂度的基本概念。

1.2.1　算法的复杂度

评价一个算法通常从时间复杂度和空间复杂度两个方面考虑。分析算法的时间复杂度时,并不是要得到算法运行所需要的时间,而是要得到一个估计量。另一方面,运算时间只能依靠实际实验得到,因此,运算时间依赖于计算环境,用运算时间衡量复杂度意义不大。算法的运行时间与算法中语句的运算数呈正比例,如果运算数多,则运行时间就长。算法中的语句运算数称为时间频度,记为 $T(n)$,其中 n 是问题规模。

算法的时间复杂度是时间频度 $T(n)$ 的渐近估计量。称算法的复杂度为 $O(T(n))$,如果存在正常数 n_0 和 c 使得任意 $n>n_0$,其复杂度都小于 $cT(n)$。也就是说,算法的复杂度与 $T(n)$ 具有相同数量级,其中 $T(n)$ 为 n 的函数。

- 多项式时间算法:称算法为多项式算法,如果其复杂度为 $O(p(n))$,其中 $p(n)$ 是 n 的多项式。
- 指数级时间算法:称算法为指数级算法,如果其复杂度为 $O(c^n)$,其中常

数 $c>1$。

如果一个算法是指数级时间算法，则算法的复杂度随着问题规模呈指数级增长。

算法的空间复杂度是指算法在计算机内执行时所需存储空间。称算法的空间复杂度 $S(n)$ 为 $O(S(n))$，即算法的复杂度与 $S(n)$ 具有相同数量级，其中 $S(n)$ 为 n 的函数。

1.2.2 问题的复杂度

P-类问题：如果存在一个多项式时间算法，或该问题能由确定型图灵机在多项式时间内解决，称该问题为 P-类问题。

NP-类问题：如果至今没有找到多项式时间算法解的一类问题，或该问题能由非确定型图灵机在多项式时间内解决，称该问题为 NP-类问题。

NP-完全问题(NPC)：此类 NP 问题中的所有的 NP 问题都可以用多项式时间归约到某一个 NP-完全问题。它是 NP 类中"最难"的问题，换言之，它们是最可能不属于 P 类的。

NP-hard 类问题：若 NP 中所有问题到该问题是图灵可归约的，称该问题为 NP-hard 类问题。对于这一类问题，一般认为不存在多项式时间的精确性算法求得最优。

迄今为止，人们还没有证实或证伪 P≠NP? 图 1-1 给出了在 P≠NP 条件下，P、NP、NPC 和 NP-hard 的关系。

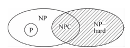


图 1-1 P、NP、NPC 和 NP-hard 关系图

本书涉及的旅行商问题、多维背包问题、定向问题、属性约简、卫星资源调度问题都是 NP-hard 类问题。

1.3 智能优化方法概述

面对形形色色的优化问题，人们已经提出了大量的优化算法。优化算法可分为精确算法和近似算法。

- 精确算法：精确算法以找到问题最优解为目标。典型的精确算法包括动态规划、分支定界算法、割平面法等。

- 近似算法：近似算法不能保证得到最优解，通常以得到问题的满意解为目标。近似算法有逼近算法和启发式算法。逼近算法能够给出所得到的解质量估计以及运行时间的界。启发算法能找到大规模算例的好解，它能以可以接受的计算开销得到可以接受的解，但是不能得到解的质量估计。启发算法可分为专门的启发算法和智能计算方法。专门的启发算法是针对某一个问题设计的启发式算法；而智能计算方法能应用于几乎所有的优化问题。

在设计优化算法时，一种思路是针对问题的特殊结构，设计出专用型算法。例如单纯形算法是运筹学的经典之作，但是它只能用于线性规划问题。

经典数学规划中的算法基本是专用型算法。这类算法一般都有很好的理论支撑，强调对问题结构的数学化应用，往往能找到最优解。这当然限制了这类算法的应用范围，并且这类算法对于使用者的编程能力也提出了很高的要求。

随着人类社会的发展，新的复杂问题层出不穷，许多问题要求人们在较短的时间内得到一个满意的解，因此迫切需要通用性强且易于实现的算法。

这些困境促使人们探索新的优化方法。同时，随着科学技术的发展，人们将生物学、物理学、化学中的原理方法与优化相结合，提出了许多智能优化方法。

1.3.1　常用的智能优化方法

1. 遗传算法

1）简介

遗传算法[41]是最早的智能计算方法。它模拟自然界进化现象，借鉴达尔文的进化论以及生物遗传学的基本思想求解优化问题。

1859年，达尔文在其经典巨著《物种起源》中提出了自然选择（天择）学说。其核心思想是“物竞天择，适者生存”。具体地说，生物间存在生存争斗，适应者存活下来，不适应者被淘汰。生物通过遗传、变异和自然选择，从低级到高级，从简单到复杂进化发展。现代生物学表明：遗传物质是基因，遗传性状由基因控制，物竞天择，竞的是基因。在生物进化过程中，遗传信息保存在基因中，而染色体是基因的载体，突变和基因重组可以得到新的基因，从而产生新的染色体。而具有优良染色体的个体能更好地适应环境。J. Holland正是借鉴这一思想提出了遗传算法。在1975年出版的著作《自然与人工系统的自适应》中，他系统提出了遗传算法，还建立了模式理论（Schema Theorem），以解释遗传算法的工作机理。模式理论是遗传算法的

重要理论基石。

在遗传算法提出之初，它主要应用于自适应系统。凭借J. Holland的学生K. DeJong和D. Fogel等的大量研究，遗传算法才被用于求解优化问题。1985年，D. Goldberg在其博士论文中研究了面向天然气管道优化的遗传算法。1989年，D. Goldberg出版了《搜索、优化于机器学习中的遗传算法》，这是遗传算法发展过程中的又一里程碑式的著作。进入20世纪90年代以后，遗传算法得到了科学界和工业界的广泛重视，出现了大量的研究成果和成功的工程实践。

在数学规划中，首先用数学语言描述优化问题，建立数学模型，再利用数学中的运算求解问题。在用遗传算法求解优化问题时，首先用生物学语言描述优化问题，将解编码成一个染色体，利用目标函数和约束函数建立染色体适应度函数评价染色体的适应能力，再利用生物进化过程模拟算法求解过程。

2）基本框架

遗传算法模拟生物学的进化过程。它首先对优化问题的解进行编码，将其表达成染色体，并产生由一定数量染色体组成的初始种群。再利用遗传操作对种群中的染色体进行操作，产生出新的种群，优者繁殖，劣者淘汰，使新种群更适应于环境。末代种群中的最优个体经过解码，输出遗传算法得到的最优解。

遗传操作包括3个基本遗传算子(genetic operator)：选择(selection)、交叉(crossover)、变异(mutation)。它们都是以随机规则进行操作的，其效果与编码方法、适应度函数、参数设定等密切相关。

遗传算法的主要步骤如下：

(1) 种群初始化：进化代数计数器t设为0，随机生成N个个体作为初始种群$P(0)$，并计算种群$P(t)$中每个个体的适应度。

(2) 选择运算：依据种群中个体的适应度，将选择算子作用于种群。选择的目的是把种群中适应度高的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。

(3) 交叉运算：将交叉算子作用于种群。交叉的作用是将两个父代个体的部分基因加以交换重组而生成新个体。在遗传算法中，交叉算子发挥着重要的作用，它是产生新解的重要算子。

(4) 变异运算：将变异算子作用于种群。该算子改变种群中个体的若干基因座上的基因值。

(5) 种群更新：种群$P(t)$经过选择、交叉、变异运算之后得到下一代种

群 $P(t+1)$。

(6) 停止条件判断：$t=t+1$。若进化次数 t 等于最大允许进化代数 $\mathrm{Max}T$，则对进化过程中所得到的具有最大适应度个体进行解码，并将其作为最优解输出，算法停止。

在设计遗传算法时，需要重点考虑以下算法要素：

(1) 解的编码：编码是一种映射，它将优化问题的解空间中的元素转换成遗传空间的由基因按一定结构组成的染色体或个体。因此，编码也可以称作问题的表示。

(2) 适应度函数定义：进化论认为适者生存。这里的"适者"是指适应环境的个体。个体的适应环境能力用适应度衡量。与之相对应，遗传算法利用适应度函数评价种群中的个体的优劣程度，其定义依赖于待求解问题的目标函数。

(3) 种群初始化：在遗传算法开始迭代时，需要初始化产生一个种群。在这一过程中，需要考虑种群多样性、种群中个体的可行性、初始化过程的计算效率。一般采用随机法初始化种群。

(4) 选择算子：选择操作依据种群中个体的适应度，从种群中选择适应度高的个体，淘汰适应度低的个体进行交叉和变异。最常用的选择方法是轮盘赌选择法(roulette wheel selection)。

轮盘赌选择法来源于博彩游戏中的轮盘赌，轮盘由圆盘和指针组成。图 1-2 给出了一个 6 扇区的轮盘。每个扇区中的数字表示该扇区的面积占总轮盘的比例。

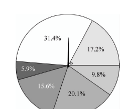


图 1-2 具有 6 个个体选择的轮盘

在轮盘赌选择法中，每个个体的选择概率是其适应度和种群中所有个体的适应度总和的比值。设种群规模为 N，个体 i 的适应度为 Fit_i，则个体 i 被选择的概率为

$$P_i = \frac{Fit_i}{\sum_{k=1}^{N} Fit_k} \tag{1-2}$$

由式(1-2)可见,个体适应度越高,其被选择的概率就越大。在计算出每个个体的选择概率后,采用轮盘赌法进行选择。将轮盘分成 N 个扇区,每个扇区对应一个个体,扇区中的数据即为个体的选择概率。每次选择,转动一次轮盘的指针,当轮盘停止转动时,指针指向的个体就被选择。这样进行 N 次后,得到规模为 N 的种群。

(5) 交叉算子:依据交叉率将种群中的两个个体按照一定的随机规则交换某些基因,能够产生新的染色体。

(6) 变异算子:一定的变异概率对种群中染色体的一个或多个基因座的基因值进行变动。

(7) 算法结构:在设计遗传算法时,种群更新方式有代(generational)方式和稳态(steady-state)方式两类。前一种方式更新整个种群,而后一种方式仅替换一些适应度低的个体。稳态方式仅更新种群中的最劣个体,如果该个体比当前得到的新个体差,则用新个体替换最劣个体。

2. 模拟退火算法

1983 年,Kirkpatrick 提出了模拟退火算法[42]。顾名思义,这一算法模拟热力学中的退火过程,采用 Metropolis 准则接受新的解或状态,避免陷于局部最优。作为一种全局优化算法,它已经用于求解连续问题、组合优化问题等不同类型的问题,在生产调度、控制工程、机器学习、神经网络、图像处理等领域得到了广泛的应用。

1) 基本原理

退火过程是一个典型的物理过程。它包括加温过程、等温过程和冷却过程。金属物体在退火后会变得柔韧。图 1-3 给出了在每个过程中物体的宏观和微观变化特点。一般地,退火过程中的温度是逐渐下降的。如果高温金属物体的温度急剧下降,它没有达到平衡态,而是处于非均匀的亚稳态,这就是淬火过程。经过淬火过程,物体能量并没有达到最小值,它能提高金属的强度和硬度,但会减弱韧度。

工业上,为了使材料满足一定的硬度和塑性要求,去除残余应力,得到预期的物理性能,通常对材料进行退火处理。金属物体的退火过程实际上就是将固体加温至充分高,再让其徐徐冷却。在加热金属的过程中,金属原子的热运动不断增强,内能增大,原子的有序稳定的状态被破坏,呈随机无序的状态。而在冷却的过程中,随温度的缓慢降低,金属原子由高能无序的

加温过程	等温过程	冷却过程
宏观特点： 1) 固体将溶解为液体； 2) 熵增加	宏观特点： 系统处于平衡态	宏观特点： 1) 液体转变为固体； 2) 熵减少
微观特点： 1) 分子的热运动增强； 2) 分子分布从有序转变为无序	微观特点： 分子的热运动减弱	微观特点： 1) 分子的热运动减弱； 2) 分子分布变为有序

图 1-3　退火的 3 个过程

状态趋于低能有序。为了使金属原子在每一个恒定温度下都能处于一个较低能量的状态即达到充分的热平衡，冷却过程温度必须缓慢降低。这个过程可以用蒙特卡罗方法模拟，该方法虽然比较简单，但需要大量采样才能获得比较精确的结果，计算量较大。

鉴于物理系统倾向于能量较低的状态，而热运动又妨碍它准确落到最低态的物理形态，采样时只需着重取那些有贡献作用的状态就可较快达到较好的效果。1953 年，Metropolis 等受到蒙特卡罗模拟方法的启示，提出了一种重要性采样法，即以概率接受新状态。在温度 t，从当前状态 i 的邻域随机产生新状态 j，两者的能量分别为 E_i 和 E_j，若 $E_i > E_j$，则接受新状态 j 为当前状态；否则，依据以下概率

$$p_t = \exp\left(-\frac{E_j - E_i}{Kt}\right) \tag{1-3}$$

接受状态 j，其中 K 为玻耳兹曼(Boltzmann)常数，$\exp(x)$ 为变量 x 的指数函数。当这个过程多次重复，即进入大量迁移后，系统将趋于能量较低的平衡态。各状态的概率分布将趋于一定的正则分布。这种接受新状态的方法被称为 Metropolis 准则，它能够大大减少采样的计算量。

从 Metropolis 准则中可以看出，高温下可接受与当前状态能量差较大的新状态，而在低温下只能接受比当前能量低的新状态或与当前能量差较小的新状态。这与不同温度下的热运动影响金属原子重新排列的过程一致。在温度趋于零时，就不能接收任一个 $E_j > E_i$ 的新状态 j。

1983 年，Kirkpatrick 等根据金属物体的退火过程与组合优化问题之间存在的相似性和 Metropolis 等提出的原始模拟退火算法，提出了现代版本的模拟退火算法，并且成功利用它解决了许多组合优化问题。

在模拟退火算法中，组合优化问题的一个解相当于退火过程中物体的一个状态，该解对应的目标函数相当于该状态的能量函数，而最优解就对应于最低能量的状态。算法设定一个初始高温，按照 Metropolis 准则在当前最优解的领域中搜索有潜力的新解，通过控制温度参数 t 的下降模拟温度徐徐降低的冷却过程。模拟退火算法在每个温度下进行多次的搜索，趋于最优解模拟物体在每个温度下都充分地达到热平衡，而最终趋于能量最小的基态。

2）算法基本要素及过程

模拟退火算法的基本思想是：将待优化问题的可行解看作是物体的原子状态，将目标函数看作物体原子的能量函数，模拟物体退火过程中粒子的热运动。降温过程中，结合概率突跳特性的 Metropolis 抽样准则，在解空间中随机搜索全局最优解，在陷入局部最优时，能以一定概率跳出，最终趋于全局最优解。

在模拟退火算法执行过程中，算法的效果取决于一组控制参数的选择，关键技术的设计对算法性能影响较大。

本节从算法使用的角度讨论算法实现的一些要素，包括状态表示、邻域定义、热平衡达到和降温控制等的概念。

（1）状态表达，在模拟退火算法求解优化问题时，一个状态对应于一个解。为了有效求解优化问题，应采用合适的编码表示解或状态。例如，在求解背包问题时，可采用 0-1 编码；求解旅行商问题时，可采用顺序编码；对于连续函数优化，可以采用实数编码。

（2）邻域定义，模拟退火算法从一个解出发，探索其邻域，寻找改进解。邻域决定搜索范围。

模拟退火算法的邻域定义与局部搜索中的定义一样。

例如在车辆路径问题中，一种邻域定义是采用一系列 k-边交换操作作为邻域，k 表示交换边的个数，k 越大，其邻域解与当前解差别越大。

模拟退火算法采用基于 Metropolis 准则的邻域移动方法。

如果一个新解优于当前解，当前解被新解替换，那么就称从当前解移动到这个新解，否则，依据一定的概率决定当前解是否移向新解。

Metropolis 准则中状态转移概率 p_{ij} 定义如下：

$$p_{ij} = \begin{cases} 1, & f(j) \leqslant f(i) \\ \exp\left(-\dfrac{f(j)-f(i)}{KT}\right), & \text{其他} \end{cases} \tag{1-4}$$

其中 i 为当前状态，j 为其邻域中的一个状态，其目标函数值分别为 $f(i)$ 和 $f(j)$，当前温度为 T。可见，当抽样得到的邻域新解优于当前解时，无条件



移动；当新解劣于当前解时，以一定概率移动。

从 Metoropolis 准则中可以看出，当 T 很大时，状态转移概率趋于 1，当前邻域中的任一状态都可能被接受，此时算法正在进行全局搜索；而当 T 很小时，状态转移概率趋于 0，它只会接受当前邻域中更好的状态，此时算法进行局部搜索。

因此，模拟退火算法既具备跳出局部优解的能力又具备探索全局最优的能力。

（3）热平衡达到，工业退火要求温度缓慢下降，以保证金属原子在每个温度下达到稳定的能量态。但是在实际应用中达到理论的平衡状态是不可能的。模拟退火算法采用在每一个温度，在邻域中寻找一定数量的解，这一循环称为算法的内循环。内循环次数应该尽可能大，以达到近似热平衡过程。内循环次数的选择与问题的解空间大小有关。当解空间较大时，可以将内循环次数设为较大的数。另外，也可以依据其他条件动态调节内循环次数。例如，高温时迭代次数减少，低温时迭代次数增加。

（4）降温函数。

模拟退火算法的搜索能力与退火速度（温度的降低速度）密切相关。

在高温状态下，当前邻域中几乎所有的解都会被接受，算法进行全局搜索；当温度变低时，当前邻域中越来越多的解将会被拒绝，算法进行局部搜索。

在同一温度下，需要对邻域进行充分的搜索以达到热平衡状态。如果温度下降速度太快，则可能错过最优解，过早地陷入局部最优；如果温度下降速度太慢，又可能会降低算法的收敛速度。因此，降温函数的选择对于模拟退火算法的性能有重要影响。

3. 禁忌搜索

禁忌搜索[42]是 Fred Glover 提出的一类智能计算方法。禁忌是人类处理复杂难题时避免迂回犯错的重要策略之一。早在 1977 年，Fred Glover 已经开始探索利用这一思想设计算法。直到 1986 年才正式提出禁忌搜索。

利用禁忌表避免迂回搜索，并利用藐视规则接受劣解实现全局优化。在禁忌搜索提出以前，人们一般利用局部搜索求解复杂的优化问题。但是局部搜索最终停止于一个局部最优点。禁忌搜索是对局部邻域搜索的扩展，它提供了一种逃离局部最优解的有效方法。

1）基本思想

禁忌搜索是局部搜索的一种改进。在局部搜索中，它首先构造一个可行解 x 作为当前解（incumbent solution），在一定的邻域 $N(x)$ 内进行贪婪

搜索,找到当前邻域内的最优解 x'(即局部最优解),再以此最优解为新的当前解进行邻域搜索,重复上述过程,直到找到满足条件的解。邻域搜索的优势在于算法结构简单、容易实现,但搜索结果依赖初始解,且只能搜索到邻域内的局部最优解。

为了在搜索过程中避免重复且能够跳出局部最优解,从而实现全局搜索,禁忌搜索算法采用类似于人类短期记忆的禁忌表,将算法搜索过程中最近的若干次移动加入禁忌表中,禁止在之后的迭代中进行移动,避免重复搜索已经搜索过的邻域。同时,禁忌表使算法能够接受劣解,将算法带入新的区域进行搜索。

算法循环过程中会不断更新禁忌表,在一定次数的循环后,最早进入禁忌表的移动会从禁忌表中删除,这就是所谓的"解禁"。

在搜索过程中,某些处于禁忌表中的移动有可能会使邻域搜索得到优于当前最优解的解,因此禁忌搜索又提出了渴望水平概念,当移动达到渴望水平,不论它是否在禁忌表中,都会被接受,即"破禁"。

禁忌搜索中禁忌表和渴望水平是最重要的两部分,可以使算法跳出局部最优,在一定的迭代次数下得到一个相对较优的解。

2) 算法要素

设计禁忌搜索算法时,需要考虑以下基本要素:初始解、搜索空间和邻域结构、适值函数、禁忌表、选择策略、渴望水平、停止准则、编码方法。

(1) 初始解。禁忌搜索算法是单点法,其性能依赖于初始解质量。当初始解质量较好时,算法能够较快收敛,且结果质量较好;反之,初始解质量较差时,会降低禁忌搜索的收敛速度。在求解一些约束条件较复杂的组合优化问题时,如果采用随机法一般难以得到质量较好的解,甚至也难以得到一个可行解。一个解决方法是:先利用问题的结构信息,应用一些简单的算法得到一个较好的初始解,再用禁忌搜索求解,从而提高算法的求解质量和效率。

(2) 搜索空间和邻域结构。搜索空间是由搜索过程中所有可能解组成的集合。例如,在车辆路径问题中,搜索空间指的是所有满足约束的解集。然而在一些问题中,定义搜索空间并不是一件简单的事,尤其是在有多种定义方式时,如何合理定义搜索空间将直接影响到算法效率。特别地,在一些问题中,搜索空间并不只包括可行解,而且允许访问不可行解有助于求解问题。邻域结构与搜索空间密切相关。在禁忌搜索中,当前解 S 的邻域结构 $N(S)$ 由其所有局部修正组成。具体地,邻域结构 $N(S)$ 针对特定的待求解问题,往往有许多邻域结构的定义方法。在应用禁忌搜索求解问题时,选择

和定义搜索空间与邻域结构是其中的关键步骤，这要求设计算法时对问题要有充分的认识和了解。

（3）适值函数。适值函数是对搜索结果的评价。一般而言，以目标函数直接作为适值函数是比较常用的方法。当目标函数的计算比较困难时，可以对目标函数进行适当的变形，作为适值函数以便于计算，从而节省计算时间，但变形后的适值函数必须是严格单调的，且适值函数的最优性与目标函数的最优性必须一致。

（4）禁忌表。禁忌表是禁忌搜索算法区别于局部搜索的关键，是禁忌搜索算法的核心。为能使算法跳出局部极小点，需要接受非改进移动。

禁忌表的作用在于避免循环，换言之，避免接受一个已经经历过的解。从数据结构上讲，禁忌表是具有一定长度的先进先出的队列。

通过避免访问以前搜索的区域，禁忌能使算法探索新的区域。在搜索过程中，禁忌对象保存在禁忌表中，而且通常只需要保存有限的信息。在设计禁忌表时，要考虑对什么禁忌以及禁忌多久。

（5）渴望水平。

禁忌搜索中，某些情况下，仅仅采用禁忌表会禁止一些可能得到高质量解的操作，甚至会导致算法停滞（即陷入局部最优）。因此，有必要采取措施以取消禁忌。即当某个移动满足某个条件时，不论该移动是否在禁忌表中，都接受这个移动，并更新当前解和当前最优解。这种使移动不受禁忌表禁忌的条件称为渴望水平，或称为特赦准则、藐视准则。

4. 粒子群算法

粒子群优化算法[43]是一种基于种群的全局搜索算法和随机搜索算法。该算法以简单、易于实现、无须梯度信息、参数少、精度高和收敛速度快等优点受到学术界和工业界的重视，成功地解决了一系列连续和离散的优化问题，成为进化计算领域的研究热点。

粒子群优化算法是由心理学研究者 James Kennedy 和计算智能研究者 Russell Eberhart 在受到人工生命研究结果的启发后于 1995 年提出的，它是模拟动物群体的智能行为和使用计算机对这些认知行为进行仿真后的产物。

自然界中的许多生物都具有一定的群体行为，揭示自然界生物的群体智能行为并通过计算机建模是人工生命研究的主要内容之一。如图 1-4 所示，科学家们通过对鸟群、鱼群等群体性行为的研究发现：单个个体的行为很简单，一般不具有规律性，但是许多个体组成的群体通常表现出某种智能行为，表现出高度的组织性和纪律性。这一发现引起了科学家们的高度关

注，吸引了包括生物科学家、计算机科学家和社会心理学家等研究的兴趣，促使他们展开广泛而深入的研究。例如，1987 年 Reynolds 使用粒子系统模拟鸟群的集体行为。1990 年，Heppner 和 Grenander 发现鸟巢能吸引鸟群。他们发现，鸟群在飞行中可以改变方向，也可以朝着某一特定方向飞行，还可以重组队形，这其中一定具有某种潜在的规律。后来，这两个发现被用于标准的粒子群算法。

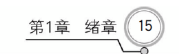


图 1-4　自然界中的鸟群和鱼群

社会心理学研究，特别是动态社会影响理论的发展，是粒子群算法发展的又一重要来源。粒子在一个问题的搜索空间中可以看作一种人类的社会行为，个体可以根据环境的变化及时地调整他们的信念和态度，从而和群体的行为保持一致。这些研究为粒子群算法的提出奠定了思想来源和理论基础。

1）基本原理

粒子群优化算法源于对鸟类觅食行为的研究。鸟类觅食时，寻找食物最简单有效的方法是搜寻目前距离食物最近的周围区域。生物群体中个体与个体之间、个体与群体之间相互影响、相互作用，体现在群体中的信息共享，促进群体的进化发展。粒子群算法是根据生物种群的智能行为得到的启发求解优化问题，每个粒子都代表优化问题在搜索空间中的一个潜在解和对应一个由适应度函数决定的适应度值，每个粒子的速度决定飞行的方向和距离，速度根据粒子自身和其他粒子的移动经验进行动态的调整，以此实现个体在可行解空间中的寻优。粒子群算法首先对可行解空间中的一群粒子初始化，每个粒子都代表极值优化问题的一个潜在可行解，该粒子的特征用位置(position)、速度(velocity)和适应度 3 个指标表示。其中，适应度值通过适应度函数计算，其大小反映了粒子的优劣程度。粒子在解空间中运动时，其位置更新是通过跟踪个体极值 P_{best} 和群体极值 G_{best} 完成的。其中，个体极值 P_{best} 是指在个体所经过的位置中通过计算其适应度值得到的最优位置，群体极值 G_{best} 是指种群中所有粒子搜索到适应度最优的位置。



粒子每更新一次位置，就计算一次适应度值，通过比较新粒子的适应度值与个体极值、群体极值的适应度值来更新个体极值 P_{best} 和群体极值 G_{best}。

一个由 n 个粒子组成的种群 $X=(x_1,x_2,\cdots,x_n)$，在 D 维搜索空间中以一定的速度飞行，每个粒子在搜索时，充分考虑搜索到的历史最好位置和种群内其他粒子的历史最优点，在此基础上进行位置（或状态、解）的变化。

相关变量定义如下：

第 i 个粒子的位置表示为 $\boldsymbol{x}_i=(x_{i1},x_{i2},\cdots,x_{iD})$；

第 i 个粒子的速度表示为 $\boldsymbol{v}_i=(v_{i1},v_{i2},\cdots,v_{iD})$；

第 i 个粒子的个体极值表示为 $\boldsymbol{p}_i=(p_{i1},p_{i2},\cdots,p_{iD})$；

种群内所有粒子的群体极值表示为 $\boldsymbol{p}_g=(p_{g1},p_{g2},\cdots,p_{gD})$。

一般来说，粒子的位置和速度都是在连续的实数空间内取值。

在每次迭代过程中，粒子通过个体极值和群体极值更新自身的速度和位置信息，其数学表达式如下：

$$\begin{aligned} v_{id}^{k+1} &:= v_{id}^{k+1}+c_1r_1(p_{id}^k-x_{id}^k)+c_2r_2(p_{gd}^k-x_{id}^k) \\ x_{id}^{k+1} &:= x_{id}^k+v_{id}^{k+1} \end{aligned} \tag{1-5}$$

其中，$d\in\{1,2,\cdots,D\}$；$i\in\{1,2,\cdots,n\}$；k 为当前的迭代次数；c_1 和 c_2 称为学习因子（也称加速系数，学习因子使粒子具有自我总结和向种群中优秀个体学习的能力，并能向着自己的历史最优位置以及种群内或邻域内的历史最优位置靠近）。通常情况下，c_1 和 c_2 取为 2；r_1 和 r_2 是[0,1]均匀分布的伪随机数。粒子速度和位置的关系如图 1-5 所示。

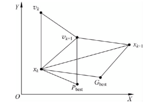


图 1-5　粒子速度与位置的关系示意图

在粒子群算法中，粒子的速度主要由 3 部分构成。

（1）粒子当前的速度，是粒子飞行中的惯性作用和其能够飞行的基本保证，表明粒子当前的状态，防止粒子大幅度改变方向，平衡粒子的全局和局部搜索能力。

(2) 认知部分,表示粒子在飞行中凭借自身的经验,向自己曾经找到过的最优位置靠近,使粒子有足够强的全局搜索能力,避免局部最优。

(3) 社会经验部分,表示粒子在飞行中考虑到社会经验,向邻域中其他粒子学习,通过借鉴和信息共享,使粒子在飞行时向邻域内所有粒子曾经找到过的最好点靠近。

2) 粒子群算法的要素

迄今为止,对于粒子群算法的研究大多以带有惯性权重的粒子群算法为基础进行分析、扩展和修正。因此,大多数文献中将带有惯性权重的粒子群算法称为标准版粒子群算法,而将前面的粒子群算法称为基本粒子群算法(或原始的粒子群算法)。

粒子群算法的要素包括算法的相关参数和算法设计中的相关问题两部分。其中相关参数包括种群大小、学习因子、最大速度、惯性权重;算法设计中的相关问题有邻域拓扑结构、粒子空间的初始化和停止条件。

在粒子群算法中,由 n 个粒子组成的集合 $X=(x_1,x_2,\cdots,x_n)$,每个粒子对应可行解空间中的一个潜在解,一个粒子从它的邻域 n_i 接收信息,其中 $n_i \in X$。标准的粒子群算法中,粒子与粒子之间的关系可以用一个无向图 $G=(V,E)$表示,其中 V 是粒子的集合;E 是边的集合,表示粒子与邻域粒子之间的配对关系。这个无向图通常被称为粒子群的拓扑结构。粒子群算法中常见的拓扑结构有星型结构、环型结构、冯·诺依曼结构和齿型结构等,如图 1-6 所示。

粒子群算法的主要流程如下:

第 1 步:初始化粒子群,随机设定位置 $\boldsymbol{x}_i$ 和速度 $\boldsymbol{v}_i$;

第 2 步:在每一次进化(迭代)中计算每个粒子的适应度值;

第 3 步:对于每个粒子 $\boldsymbol{x}_i$,如果其适应度值比所经历过的历史最好位置 P_{best}的适应度值好,则用当前位置 $\boldsymbol{p}_i$ 更新个体历史最优位置 P_{best};

第 4 步:对于每个粒子 $\boldsymbol{x}_i$,如果其历史最优适应度值比群体内或邻域内所经历的最好位置 G_{best}的适应度值好,则用当前的全局最优位置 $\boldsymbol{p}_g$ 更新种群的历史最优的位置 G_{best};

第 5 步:根据更新公式对粒子的位置 $\boldsymbol{x}_i$ 和速度 $\boldsymbol{v}_i$ 进行修正;

第 6 步:若未达到停止条件,则转到第 2 步。

5. 人工蜂群算法[44]

自然界的蜜蜂是一种社会性群居生物,在群体中,单个蜜蜂的智能与能力是有限的。然而,由一群具有简单智能的个体组成的群体却表现出令人惊讶的智能。无论所处的环境多么复杂,它们总能找到蜂巢周围、距离适中

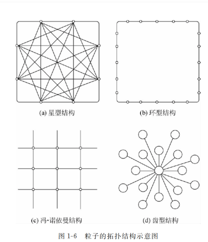


图 1-6 粒子的拓扑结构示意图

且食物最丰富的食物源。蜂群表现出的智能行为引起了科学家的极大兴趣。

2005 年,Karaboga 系统地提出了人工蜂群算法(Artificial Bee Colony Algorithm)算法,并将人工蜂群算法应用于求解函数值优化问题,取得很好的效果。

1)基本原理

Seeley 最早提出了一种蜂群的群居行为模型:自组织模拟模型。该模型中,群体由各种角色的蜜蜂组成,虽然每个角色的蜜蜂只能完成单一的任务,但蜂群中的蜜蜂通过舞蹈、气味等信息交互方式使整个蜂群能够协同完成诸如蜂巢构建、觅食等多种较为复杂的任务。

在模型中,蜂群包括 3 种基本要素:食物源、雇佣蜂和非雇佣蜂,其中非雇佣蜂包括侦察蜂和跟随蜂。具体地,这些要素描述如下:

(1)食物源,食物源的优劣程度主要依赖以下因素:食物源到蜂巢的距离;食物源的丰富程度;食物获取的困难程度等。一般地,用食物源的收益表征这些因素。

(2)雇佣蜂,也称为引领蜂。模型中雇佣蜂指正在某个食物源觅食或

已经被这个食物源雇佣的蜜蜂。其数量一般与食物源有关。它们会把这个食物源的信息，例如离蜂巢的距离和方位、食物源的收益等信息通过舞蹈的方式告知其他蜜蜂。

(3) 侦察蜂，侦察蜂通常在蜂巢周围搜索附近的食物源。一般地，蜂群中的侦察蜂数量约占整个蜂群总数的 5%～10%。

(4) 跟随蜂，跟随蜂在舞蹈区等待由雇佣蜂带回的食物源信息，它们观察雇佣蜂的舞蹈，选择自己认为满意的食物源进行跟随。

模型中，蜂群有两种基本行为模式：①引领模式，即当一只蜜蜂找到自己认为丰富的食物源时，引领其他蜜蜂到食物源处；②放弃模式，即放弃一处食物源，寻找新的食物源。

蜂巢附近的舞蹈区是蜂群中的个体进行信息交换的主要场所，也是角色转换的地方。在舞蹈区，蜜蜂通过摇摆舞的形式与其他蜜蜂交流食物源信息。在蜂群智能行为中，信息交换的过程如下：侦查蜂飞回蜂巢，并在舞蹈区进行舞蹈，舞蹈区周围的蜜蜂通过观察进行选择，一旦选定自己的食物源，蜜蜂角色转换随之发生。食物源被选择的可能性依赖于其收益率。食物源的收益率越大，其被选择的可能性也大。

在觅食行为之初，一部分蜜蜂从蜂巢出发，它们的角色是侦察蜂，在蜂巢周围进行随机搜索。当蜜蜂搜索到食物源后便进行采蜜，并记录食物源的相关信息，此时蜜蜂的角色就转变为"被雇佣者"，其余没有进行采蜜的蜜蜂，就成为"非雇佣蜂"。其中，被雇佣的蜜蜂在觅食完成后，回巢并做如下选择：

(1) 放弃已经搜索到的食物源，角色由雇佣蜂变成侦察蜂。

(2) 在舞蹈区与其他蜜蜂进行信息共享，招募更多的蜜蜂。

(3) 继续返回采蜜，而不招募其他蜜蜂。

非雇佣的蜜蜂会做出如下选择：

(1) 以侦察蜂的身份对蜂巢附近的食物源展开搜索。其搜索可以是完全随机的，也可依赖于先验知识。

(2) 选择一个雇佣蜂进行跟随，其角色转变为跟随蜂，并在食物源的邻域进行搜索。

2) 人工蜂群算法的基本过程

在用人工蜂群算法求解优化问题时，食物源对应于待优化问题的一个可行解，食物源的丰富程度(即适应度)代表解的质量。引领蜂和跟随蜂各占蜂群数量的一半，每个食物源只有一个引领蜂，即引领蜂数量等于蜜源数量。当一个食物源被放弃时，它所对应的引领蜂就变成了侦查蜂。在算法初始化后，蜜蜂开始对全部的食物源进行搜索。

引领蜂会先对全局进行搜索，并比较搜索前后食物源的丰富程度，蜜蜂会选择食物源较为丰富的目标。当所有的引领蜂完成搜索后，它们会回到信息交流区(即舞蹈区)将其了解的食物源相关信息与其他蜜蜂进行信息共享。跟随蜂则会根据引领蜂提供的信息按照一定的概率选择引领蜂进行跟随。适应度越高的食物源被选择的概率越大。然后，跟随蜂会和引领蜂一样进行邻域搜索，并选择较好的解。

1.3.2 智能优化方法的一般框架

智能优化方法是人工智能的一个重要分支。笔者将从人工智能视角，提出智能优化方法的一般框架。

人工智能的工程目标是设计制造出智能产品[45]，替代人解决问题或完成任务。在解决问题时，需要用到知识。所谓知识，是可用于解决该问题的领域信息。为了有效解决问题，需要解决以下问题：

(1) 知识表示：将知识表示成能用计算机处理的符号；

(2) 知识发现与学习：从经验中不断地自动获取知识；

(3) 知识推理与应用：利用知识产生行为。

一般而言，智能优化方法是一种基于采样的迭代过程。在求解问题时，智能优化方法将该问题进行编码。在每一代，它主要包括产生解和信息加工两个过程。产生解过程对应于知识推理与应用过程，而信息加工过程对应于知识发现与学习过程。例如经典遗传算法选择两个个体的染色体进行重组得到解，其信息加工机制是种群更新机制。在遗传算法中，知识利用染色体进行编码表示。因此，智能优化方法是符合人工智能解决问题的一般范式的。

本书将讨论的蚁群算法在知识表示、知识发现与学习及知识推理与应用方面具有显著特色。在蚁群算法中，信息素具有重要的作用，它是知识的载体。在蚁群算法演化过程中，蚁群通过信息素进行信息交互，并用以指导解的构造。

1.3.3 智能优化方法分类

目前，已有数十种智能计算方法。常见的分类准则有以下几种。

- 种群 vs 单点：单点法在搜索过程中仅对一个解进行操作和变换；而基于种群的算法对一个种群进行演化。
- 记忆的作用：有些智能计算方法是无记忆的，即在搜索过程中，没有利用到动态提取的信息，例如模拟退火算法。而有些智能计算方法

是有记忆的，它们利用到在线学习的信息，如禁忌搜索的长期和短期记忆。

- 构造型 vs 非构造型算法：通过构造过程得到解。
- 确定型 vs 随机型：在求解问题过程中，确定性的智能计算采用确定型决策。而在随机智能计算方法中，采用了许多随机规则。在确定型算法中，如果初始解给定，则输出解是确定的。而在随机型算法中，即使给定初始解，最终得到的解也可能是不同的。

本书研究的蚁群智能优化算法与粒子群算法和蜜蜂算法均属于群体智能(Swarm Intelligence)范畴[44]。群体智能是受蚁群、鸟群、蜂群等群居生物体行为启发，通过模拟这些群居生物行为产生的计算方法。在自然界中，单个蚂蚁(或鸟或蜜蜂)的能力非常有限，难以独立生存，但是一群蚂蚁却能通过相互协作很好地适应环境，表现出智能行为。

在群体智能中，一群相互之间直接或间接通信的个体组成群体，这些个体通过相互协作求解问题。

1994 年，Mark Millonas 提出了群体智能应该遵循的 5 条基本原则。

(1) 接近原则(Proximity Principle)：群体中的个体具有对空间和时间进行简单计算的能力。

(2) 质量响应原则(Quality Principle)：群体能够对环境中的质量因子作出响应。

(3) 多样性响应原则(Principle of Diverse Response)：群体的行动和响应范围不能太窄，应具有多样性。

(4) 稳定性原则(Stability Principle)：每次环境变化时，群体不应该随之改变其行为模式。

(5) 适应性原则(Adaptability Principle)：在保证计算代价的条件下，群体能够改变其行为模式。

1.3.4 智能优化方法的特点

与传统优化方法相比，智能计算方法一般具有以下特点：

(1) 自适应性强。对待求解的优化问题没有过多的要求，一般不要求满足可微性、凸性等条件，在迭代过程中，一般只用到目标函数值等信息，不必用到目标函数的导数等问题信息。这使得智能计算方法具有很强通用性。

(2) 优良的全局寻优能力。它们在解空间进行全局搜索，按照一定的机制指导搜索，算法具有很好的鲁棒性，对初始条件不敏感，具有很强的容差能力。

（3）易于实现。智能计算方法原理简单，一般不需要数学推导。

1.4　本书内容及组织

本书的第2章讲述蚁群算法的基本原理和算法要素，概述了算法的国内外研究现状。随后的8章内容总结作者近年来的研究成果(包括论文[47]～[52]以及一些实际课题)。主要内容如下：

（1）在第3章中，首先介绍旅行商问题及其相关知识，再提出有限级信息素蚁群算法，把信息素分成有限个级别，用完全不同的方式更新信息素，并且信息素的更新量与目标函数值无关。

（2）在第4章，针对多维背包问题，提出了一类自适应蚁群算法，该算法采用自适应方法选取信息素下界。其基本思想是在解的平均差异量过小时通过修正信息素下界避免停滞。针对多维背包问题的实验结果表明，提出的方法能有效地平衡多样性和强化性。

（3）在第5章和第6章，针对定向问题及团队定向问题，结合问题的特点，分别提出高效的蚁群算法。

（4）第7章提出了求解属性约简的蚁群算法。属性约简是一个重要的特征提取方法。本章提出了3种蚁群算法。依据其信息素释放方式，分别称这3种算法为边模式蚁群算法、团模式蚁群算法和点模式蚁群算法。

（5）第8章给出了卫星资源调度问题模型，采用蚁群算法进行求解。该资源调度问题源于西安卫星测控中心的实际需求。

（6）第9章将蚁群算法用于求解旅游路线规划问题，并将其与多个智能计算方法进行比较。

（7）第10章提出了一类求解多目标组合优化问题的蚁群算法，并在多目标旅行商问题和多目标多维背包问题中进行验证。多目标组合优化是运筹学的重要分支，本章将给出如何用蚁群算法求解这类问题，并分析所提出算法的性能。

参考文献

[1]　陈宝林. 最优化理论与算法[M]. 北京：清华大学出版社，2005.

[2]　康崇禄. 蒙特卡罗方法理论和应用[M]. 北京：科学出版社，2016.

[3]　王凌. 智能优化算法及其应用[M]. 北京：清华大学出版社，2001.

[4]　Russell C. Eberhart，Yuhui Shi. 计算智能：从概念到实现[M]. 北京：人民邮电出版社，2009.

[5] Dorigo M，Stützle T. 蚁群优化[M]. 张军，胡晓敏，罗旭耀，译. 北京：清华大学出版社，2007.

[6] Dorigo M，Maniezzo V，Colorni A. Ant system：Optimization by a colony of cooperating agents[J]. IEEE Transactions on System Man，and Cybernetics-Part B，1996，26：29～41.

[7] Dorigo M，Caro G Di. The ant colony optimization metaheuristic[C]. Corne D，Dorigo M，Glover F，ed. New Ideas in Optimization. London，U K：McGraw-Hill，1999：11～32.

[8] Zecchin A C，Simpson A R，Marier H R，et al. Parametric study for an ant algorithm applied to water distribution system optimization[J]. IEEE Transactions on Evolutionary Computation，2005，9(2)：175～191.

[9] Zhang N，Feng Z. Cooperative ant colony optimization for multi-satellite resource scheduling problem[C]. 2007 IEEE Congress on Evolutionary Computation. Singapore，2007：2822～2828.

[10] 金飞虎，洪炳熔，高庆吉. 基于蚁群算法的自由飞行空间机器人路径规划[J]. 机器人，2002，24(6)：526～530.

[11] 丁滢颖，何衍，蒋静坪. 基于蚁群算法的多机器人协作策略[J]. 机器人，2003，25(5)：414～418.

[12] 董玉成，陈义华. 基于蚂蚁算法的移动机器人路径规划[J]. 重庆大学学报，2003，26(3)：49～51.

[13] 樊晓平，罗熊，易晟. 复杂环境下基于蚁群算法的多机器人路径规划[J]. 控制与决策，2004，19(2)：166～170.

[14] 朱庆保. 动态复杂环境下的机器人路径规划蚂蚁预测算法[J]. 计算机学报，2005，28(11)：1898～1906.

[15] 侯云鹤，熊信艮，吴耀武，等. 基于广义蚁群算法的电力系统经济负荷分配[J]. 中国电机工程学报，2003，23(3)：59～64.

[16] Gomez J F，Khodr H M，De Oliverira P M，et al. Ant colony system algorithm for the planning of primary distribution circuits[J]. IEEE Transactions on Power Systems，2004，192(2)：996～1004.

[17] 翟海保，程浩忠，吕干云，等. 多阶段输电网络最优规划的并行蚁群算法[J]. 电力系统自动化，2004，28(20)：37～42.

[18] 程晓荣，叶显熠，梁玉泉，等. 基于改进蚁群算法的输电网络扩展规划[J]. 电力系统自动化，2006，30(20)：37～40.

[19] 樊友平，陈允平，黄席樾，等. 运载火箭控制系统漏电故障诊断研究[J]. 宇航学报，2004，25(5)：507～513.

[20] 汪镭，吴启迪. 蚁群算法在系统辨识中的应用[J]. 自动化学报，2003，29(1)：102～109.

[21] Parpinelli R S，Lopes H S，Freitas A A. Data mining with an ant colony optimization algorithm[J]. IEEE Transactions on Evolutionary Computation，2002，6(4)：321～332.

[22] Tsai C F, Tsai C W, Wu H C, et al. ACOAF: a novel data clustering approach for data mining in large database[J]. Journal of System and Software, 2004, 73(1): 133~145.

[23] Meshoul S, Batouche M. Ant colony system with external dynamics for point matching and pose estimation[J]. Pattern Recognition, 2002, 3: 823~826.

[24] 闫伟，张浩，陆剑峰. 一种离群数据挖掘新方法的研究与应用[J]. 控制与决策，2006，21(5)：563~567.

[25] Xu X H, Chen L. An adaptive ant clustering algorithm[J]. Journal of Software, 2006, 17(9): 1884~1889.

[26] 郑肇葆，叶志伟. 基于蚁群行为仿真的影像纹理分类[J]. 武汉大学学报，2004，29(8)：669~673.

[27] 冯远静. 群体协同蚁群算法及其在图像分割中的应用[D]. 西安：西安交通大学，2004.

[28] 王晓年，冯远静，冯祖仁. 一种基于主动轮廓模型的蚁群图像分割算法[J]. 控制理论与应用，2006，23(4)：515~522.

[29] 王和平，柳长安，李为吉. 基于蚁群算法的无人机任务规划[J]. 西北工业大学学报，2005，23(1)：98~101.

[30] 李士勇，杨丹. 基于改进蚁群算法的巡航导弹航迹规划[J]. 宇航学报，2007，28(4)：903~907.

[31] 罗德林，段海滨，吴顺详，等. 基于启发式蚁群算法的协同多目标攻击空战决策研究[J]. 航空学报，2006，27(6)：1166~1170.

[32] 李亮，迟世春，林皋. 基于蚁群算法的复合行法及其在边坡稳定分析中的应用[J]. 岩土工程学报，2004，26(5)：691~696.

[33] 贺益君，陈德钊. 连续约束蚁群优化算法的构建及其在丁烯烷化过程中的应用[J]. 化工学报，2005，56(9)：1708~1713.

[34] 何莲莲，石峰，周怀北. 改进的蚁群算法在2DHP模型中的应用[J]. 武汉大学学报，2005，51(1)：33~38.

[35] 霍军周，李广强，腾弘飞，等. 人机结合蚁群/遗传算法及其在卫星舱布局设计中的应用[J]. 机械工程学报，2005，41(3)：112~116.

[36] 段海滨，王道波，黄向华，等. 基于蚁群算法的PID参数优化[J]. 武汉大学学报，2004，37(5)：97~100.

[37] 段海滨，王道波，于秀芬. 基于改进蚁群算法的飞行仿真转台的控制优化[J]. 中国空间科学技术，2007，28(4):36~43.

[38] 李士勇. 蚁群算法及其应用[M]. 哈尔滨：哈尔滨工业大学出版社，2004.

[39] 段海滨. 蚁群算法原理及其应用[M]. 北京：科学出版社，2005.

[40] 戈德赖希. 计算复杂性[M]. 北京：人民邮电出版社，2010.

[41] 王小平，曹立明. 遗传算法——理论、应用与软件实现[M]. 西安：西安交通大学出版社，2002.

[42] 汪定伟. 智能优化方法[M]. 北京：高等教育出版社，2007.

[43] 张军. 计算智能[M]. 北京：清华大学出版社，2009.

［44］ 段海滨. 仿生智能计算[M]. 北京：科学出版社，2011.
［45］ David L. Poole，Alan K. Mackworth. 人工智能：计算 agent 基础[M]. 北京：机械工业出版社，2015.
［46］ Millonas M M. Swarms，phase transitions，and collective intelligence［M］. Langton C G，ed. Artificial Life Ⅲ，Reading，MA：Addison-Wesley Publishing Co，1994.
［47］ 柯良军，冯祖仁，冯远静. 有限级信息素蚁群算法[J]. 自动化学报，2006，32(2)：296～303.
［48］ Liangjun Ke，Zuren Feng，Zhigang Ren，et al. An ant colony optimization approach for the multidimensional knapsack problem[J]. Journal of Heuristics，2010，16(1)：65～83.
［49］ Liangjun Ke，Claudia Archetti，Zuren Feng. Ants can solve the team orienteering problem[J]. Computers and Industrial Engineering，2008，54(3)：648～665.
［50］ Liangjun Ke，Zuren Feng，Zhigang Ren. An efficient ant colony optimization approach to attribute reduction in rough settheory［J］. Pattern Recognition Letters，2008，29(9)：1351～1357.
［51］ Liangjun Ke，Qingfu Zhang，Roberto Battiti，MOEA/D-ACO：A Multiobjective Evolutionary Algorithm Using Decomposition and Ant Colony［J］. IEEE Transactions on Cybernetics，2013，43(6)：1845～1859.
［52］ Zhang N，Feng Z，Ke L J. Guidance-solution based ant colony optimization for satellite control resource scheduling problem[J]. Applied Intelligence，2011，35(3)：436～444.



蚁群优化方法概述

2.1 蚁群算法的思想起源

蚂蚁作为一种社会昆虫,其食物搜索行为具有非常高的协作性。研究发现,当蚂蚁在蚁穴和食物源之间行走时,会释放一种称为信息素的物质,这些物质形成一条指示轨迹。由于蚂蚁在较长路径上行走时需要更多的时间,而信息素是随着时间挥发的,因此较短路径上的信息素强度(intensity)高于较长路径上的强度。蚂蚁可以感知到环境中这种物质的存在及其强度,并在移动时,倾向于选择信息素强度高的轨迹。最终它会从初始的随机路径搜索,逐步稳定到最短路径上来。蚂蚁对环境中的刺激物作出反应,并产生新的刺激物,这些刺激物既作用于自己,也对群体中的其他个体产生影响。这种间接的通信方式形成了一种行为协作方式。它具有两个突出特点:一是通信个体通过释放信息素修正其对所处环境的物理状态;二是信息素只能被访问该局部状态的通信个体感应到。这种以环境状态的物理修正为媒介,而且只能被局部接收的通信方式称为 Stigmergy。

1989 年,Gross 等[1]利用一群蚂蚁做了如下实验:在蚁穴与食物源之间架设二分支桥。每次移动时,蚂蚁只能选择两个分支之一往返于蚁穴与食物源。实验观察发现,在很短时间内,大多数蚂蚁将会选择路径较短的桥,并且蚁群选择短分支的概率随着两个分支之间的长度比例增加而增大。

随后，Dorigo 等[2]做了如图 2-1 所示的经典实验。图中 A 是蚁穴，E 是食物源，HC 为一障碍物。蚂蚁在没有障碍的时候，可以通过如图 2-1(a)所示路径到达食物源。但是由于障碍的存在，蚂蚁只能经 H 或 C 由 A 到达 E(图 2-1 (b))，其路径选择受先前经过的蚂蚁留下的信息素强度的影响。右边的路径短，蚂蚁在其上留下的信息素强度高，该路径被其他蚂蚁选择的概率就高。随着信息素的积累，蚂蚁趋向于选择右边的路径(图 2-1(c))。

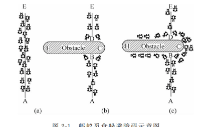


图 2-1　蚂蚁觅食躲避障碍示意图

Dorigo 等将真实蚂蚁的行为模型化，用加权弧段表示实际的路径，其中权值如图 2-2(a)所示。设 $t=0$ 时刻有 30 只蚂蚁由 A 到达 B，另有 30 只蚂蚁由 E 到达 D，蚂蚁每次经过后留下的信息素强度为 1。为方便起见，设该物质经过一个时间单位后完全挥发。在初始时刻，由于路径 BH、BC、DH、DC 上均无信息素存在，位于 B 和 D 的蚂蚁随机地选择路径。从统计的角度可以认为它们以相同的概率选择 BH、BC、DH、DC(图 2-2(b))。经过一个时间单位后，在路径 BCD 上的信息素量是路径 BHD 上信息素量的两倍。

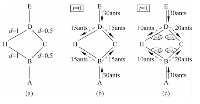


图 2-2　模型化距离

在 $t=1$ 时刻，将有 20 只蚂蚁由 B 和 D 到达 C，有 10 只蚂蚁由 B 和 D 到达 H(图 2-2(c))。随着时间的推移，蚂蚁会以越来越大的概率选择路径 BCD，最终完全选择路径 BCD，从而找到由蚁穴到食物源的最短路径。

受真实蚂蚁觅食行为的启发，Dorigo 等开创性地提出了蚁群算法，其基本思想是利用一群人工蚂蚁模拟真实蚂蚁的觅食行为来求解问题。人工蚂蚁和真实蚂蚁既有联系又有区别[3]，通过比较两者的特点，有助于了解蚁群算法的工作机理，对蚁群算法的研究也有一定的指导意义。

人工蚂蚁绝大部分行为特征来源于真实蚂蚁，它们的共同特征主要有以下 5 点。

1）人工蚂蚁和真实蚂蚁都是一群相互合作的个体

它们可以通过全局范围内相互合作找出问题较好的解。虽然每只人工蚂蚁构造可行解的行为往往比较复杂，但是高质量的解也是整个蚁群合作的结果。

2）人工蚂蚁和真实蚂蚁都通过信息素进行间接通信

真实蚂蚁在经过的路径上留下信息素，而人工蚂蚁改变其经过路径上存储的数字信息，该信息记录了蚂蚁当前解和历史解的性能状态，而且可以被以后经过该路径的蚂蚁读写。蚁群通过这种方式改变了当前人工蚂蚁所经过的路径周围的环境，同时也改变了整个蚁群所存储的历史信息。另外，蚁群算法中还有一个信息素挥发机制，它模拟真实信息素挥发，逐渐改变信息素。挥发机制使得人工蚂蚁和真实蚂蚁一样可以逐渐忘记历史遗留信息，从而使人工蚂蚁在进行路径选择时不局限于先前蚂蚁所留下的经验。

3）人工蚂蚁和真实蚂蚁都利用了正反馈机制

人工蚂蚁受真实蚂蚁觅食行为的启发，利用了正反馈机制(自催化机制)和解的隐性评价(implicit solution evaluation)[3]。解的隐性评价意味着路径越短可以越快地被人工蚂蚁经过，从而较短的路径可以接收更多的信息素。而正反馈意味着路径越短则信息素越多，从而被更多的蚂蚁选取的可能性越大。如果合理利用，正反馈可以成为基于群体的优化算法的一个有效机制。但是，在应用正反馈时应避免早熟收敛。

4）人工蚂蚁和真实蚂蚁都有一个共同的任务，并且只能局部移动

人工蚂蚁与真实蚂蚁有共同的任务，即寻找连接起点(蚁穴)到终点(食物源)的最短(最小代价)路径。真实蚂蚁不能跳跃，它们只能在相邻区域内行走；而人工蚂蚁也只能一步步地沿着问题的相邻状态移动。

5）人工蚂蚁和真实蚂蚁都采用随机的、近视的(myopic)状态转移策略

与真实蚂蚁类似，人工蚂蚁在构造解时按照一种随机决策策略在相邻

状态间移动，而且它们只是利用了局部信息而没有利用前瞻性预测未来状态。因此，它们所采用的策略在时间和空间上都是局部的。这个策略既与问题特有先验信息有关，又与由蚂蚁引起的环境局部改变有关。

人工蚂蚁还具有真实蚂蚁所不具备的行为特征，主要有以下几点：

(1) 人工蚂蚁生活在一个离散空间中，它们的移动是由一个离散状态到另一个离散状态的跃迁；而真实蚂蚁是在一个连续的空间爬行。

(2) 人工蚂蚁具有一定的记忆能力，它能保存蚂蚁过去的状态。而真实蚂蚁没有表现出这方面的能力。

(3) 人工蚂蚁释放信息素的时机可以依据具体问题而定，而真实蚂蚁是在移动的同时释放信息素。

(4) 为了提高系统的优化性能，人工蚂蚁可以增加一些额外的本领，如预测能力、局部优化、回退等。而这些本领是真实蚂蚁所不具备的。

总之，蚁群算法是真实蚁群觅食行为的一种抽象。同时，为了能有效解决实际优化问题，它赋予了人工蚂蚁一些真实蚂蚁所不具备的本领。

2.2 蚁群算法的基本框架

在文献[4]中，Dorigo 和 Caro 给出了一个针对组合优化问题的蚁群算法基本框架。组合优化问题通常可以由三元组(S, f, Ω)表示，其中 S 是候选解的集合；f 是目标函数，对于任意 $s \in S$ 有目标函数值 $f(s)$；Ω 是约束条件集合。其优化目标是寻找全局最优可行解。

组合优化问题(S, f, Ω)可以映射为具有如下特征的问题：

(1) 有限集合 $\zeta = \{c_1, c_2, \cdots, c_{NC}\}$ 表示优化问题的组成元素(component)。

(2) 根据所有的可能序列 $x = \langle c_i, c_j, \cdots, c_k, \cdots \rangle$ 定义问题的有限状态集合 χ，其中 $c_i, c_j, \cdots, c_k$ 是 ζ 的元素。序列的长度定义为 $|x|$，表示序列中元素的个数，序列长度的最大值 $l < \pm\infty$。

(3) 候选解集合 S 是有限状态集合 χ 的子集，即 $S \subseteq \chi$。

(4) 可行状态集合 $\tilde{\chi} \subseteq \chi$，由满足约束条件集合 Ω 的序列 $x \in \chi$ 构成。

(5) 非空集合 S^* 是最优解组成的集合，$S^* \subseteq \tilde{\chi}$ 且 $S^* \subseteq S$。

通过以上描述，组合优化问题可以通过图的形式表示为 $G = (\zeta, L)$，其中，ζ 表示结点集合，L 表示所有结点的连接弧集合。从而，待求解的问题转化为在一个图中搜索最小代价路径问题。问题的解对应于 G 上的可行序列(可行路径)。它的最优解即为 G 上满足约束条件的最短路径，即目标函数

的最优解。

蚁群算法是一个迭代算法，在每一次迭代中执行如下操作：一群蚂蚁同步或异步地在问题的相邻状态之间移动，它们利用关联在每个状态中的信息素和启发信息，采用状态转移规则选择移动方向，逐步构造出问题的可行解；在每只蚂蚁构造解时，可以局部地更新信息素；在所有蚂蚁都完成了解的构造之后，依据获得的解对信息素全局更新。这个迭代过程持续到某个停止条件被满足为止。常用的停止条件有最大运行时间或者允许构造解的最大数目等。用不同的方法对蚁群算法总框架的各部分进行具体化，可以产生不同的蚁群算法。图 2-3 为蚁群算法的标准流程图。

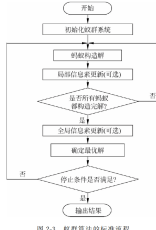


图 2-3　蚁群算法的标准流程

2.3　基本蚁群算法及其典型改进算法

蚂蚁系统（AS）是最早的蚁群算法，也是基本蚁群算法。此后，人们提出了许多改进的蚁群算法，包括蚁群系统（ACS）[5]、基于排序的蚂蚁系统[6]、最大最小蚂蚁系统（MMAS）[7]等。它们与 AS 的主要区别在于状态

转移规则和信息素更新规则的不同。其中，ACS 和 MMAS 是两类应用广泛的蚁群算法。本节先介绍 AS，然后介绍 ACS 和 MMAS 的工作原理。其他算法参见文献[8,9]。

为便于说明这些算法的基本原理，首先介绍旅行商问题（TSP）。TSP 的图论描述如下：给定 n 个结点（每个结点代表一个城市）和连接结点的弧段（边）的集合 E，对任意 $i\in N, j\in N, (i,j)\in E$，用 d_{ij} 表示弧段的长度，即城市 i 和 j 之间的距离，TSP 问题就是在图 $G=(N,E)$ 中找一条最短的 Hamilton 回路，即闭回路，经过每个结点一次且只一次，其长度为组成它的各弧段长度的和。在蚁群算法发展过程中，TSP 常用作测试问题。

2.3.1 基本蚁群算法

在基本蚁群算法中，人工蚂蚁具有以下特征：

(1) 蚂蚁在城市 i 根据信息素和启发信息选择下一个城市 j。

(2) 在从城市 i 到城市 j 移动过程中或是在完成一次循环后，蚂蚁在边 (i,j) 上释放一定量的信息素 $\tau(i,j)$。

(3) 为了满足问题的约束条件，在一次循环中，不允许蚂蚁选择已经访问过的城市。

下面给出基本蚁群算法的状态转移规则和信息素更新规则。

1) 状态转移规则

在初始时刻，蚂蚁随机选取一个结点，然后蚂蚁从一个结点移动到另一个结点，直到经过所有的结点。设第 k 只蚂蚁当前所在的结点为 i，则从该结点移到结点 j 的概率为

$$p_{ij}=\begin{cases}\dfrac{\tau(i,j)^{\alpha}\cdot\eta(i,j)^{\beta}}{\sum\limits_{u\in N_k^i}\tau(i,u)^{\alpha}\cdot\eta(i,u)^{\beta}}, & j\in N_k^i\\ 0, & \text{其他}\end{cases}\tag{2-1}$$

其中 $\eta(i,j)$ 是弧 (i,j) 上的启发信息。在 TSP 问题中，$\eta(i,j)$ 一般选取为弧长的倒数。N_k^i 是由所有未经过的点组成的集合，α,β 分别表示信息素和启发信息相对权重参数，它们控制 $\tau(i,j)$ 和 $\eta(i,j)$ 在决策中所占的比重。由式(2-1)可知，那些具有较多的信息素且较短的弧段，被选择的概率较大。

2) 信息素更新规则

经过 $n-1$ 次选择，蚂蚁完成一个回路，也就是问题的一个可行解。当蚂蚁原路返回时，在它所经过的弧段上留下信息素，用 $\Delta\tau_{ij}^k$ 表示第 k 只蚂蚁在弧段 (i,j) 上的释放的量。它的取值与相应的可行解 T_k 质量有关。设 L_k 表示该回路的长度，显然，L_k 越小，解的质量越好，在所经过的弧段上留下

的信息素 $\Delta\tau_{ij}^k$ 越大。任意弧段 (i,j) 上信息素的总改变量为

$$\Delta\tau(i,j)=\sum_{k=1}^{m}\Delta\tau_{ij}^k \tag{2-2}$$

其中 m 是蚂蚁数。

根据具体算法的不同，$\Delta\tau_{ij}^k$ 的表达形式可以不同。Dorigo[2] 曾给出 3 种不同的模型，分别为 Ant Cycle System、Ant Quantity System 和 Ant Density System。

在 Ant Cycle System 中，$\Delta\tau_{ij}^k$ 为

$$\Delta\tau_{ij}^k=\begin{cases}\dfrac{Q}{L_k}, & \text{第 } k \text{ 只蚂蚁在本次循环中经过弧}(i,j)\\ 0, & \text{其他}\end{cases} \tag{2-3}$$

其中 Q 为常数；L_k 表示第 k 只蚂蚁在本次循环中所走路径的长度。

在 Ant Quantity System 中，$\Delta\tau_{ij}^k$ 为

$$\Delta\tau_{ij}^k=\begin{cases}\dfrac{Q}{d_{ij}}, & \text{第 } k \text{ 只蚂蚁在本次循环中经过弧}(i,j)\\ 0, & \text{其他}\end{cases} \tag{2-4}$$

在 Ant Density System 中，$\Delta\tau_{ij}^k$ 为

$$\Delta\tau_{ij}^k=\begin{cases}Q, & \text{第 } k \text{ 只蚂蚁在本次循环中经过弧}(i,j)\\ 0, & \text{其他}\end{cases} \tag{2-5}$$

后两种算法与 Ant Cycle System 的区别在于，蚂蚁每走一步都要更新信息素的强度，而不是等到所有蚂蚁完成一个解的构造以后。在 Ant Quantity System 中，信息素强度的增量为 Q/d_{ij}，此时增量会随着城市之间距离的减小而增大。在 Ant Density System 中，从城市 i 到城市 j 的蚂蚁在路径上释放的信息素强度是一个与弧长无关的常数 Q。Ant Cycle System 利用全局信息更新信息素，它在求解 TSP 问题时性能较好。

此外，基本蚁群算法引入信息素挥发机制。设信息素的保持系数为 ρ，则信息素挥发系数为 $1-\rho$。信息素按式(2-6)调整

$$\tau(i,j)\leftarrow\rho\tau(i,j)+\Delta\tau(i,j) \tag{2-6}$$

至此一次迭代结束，进入下一次迭代。重复进行上述过程，直到满足某个停止条件则算法停止。

研究表明，基本蚁群算法具有以下优点：

(1) 具有较强的全局搜索能力。在算法中，一群蚂蚁通过相互协作来更好地适应环境，以获得更好的性能；利用蚂蚁群体而不是单只蚂蚁，使得算法找到全局最优解的概率增加；另外，使用概率规则而不是确定性规则指导搜索，使得算法有可能逃离局部最优。而传统优化算法对初值、迭代步

长较敏感,一旦陷入局部最优就很难逃离。

(2) 具有潜在的并行性。所有蚂蚁同时独立地在解空间中搜索,非常适合于并行实现,因此它本质上是一种高效的并行搜索算法。一方面蚂蚁的搜索行为是独立自主的,不需要集中控制;另一方面,即使一只或者几只蚂蚁做出不好的选择,整个蚁群系统仍然能够保持正常功能。这种分布式并行模式大大提高整个算法的运行效率和鲁棒性。

(3) 在优化过程中不依赖于优化问题本身的数学性质,如连续性、可导性以及目标函数和约束条件的精确数学描述等。

(4) 具有学习能力,在复杂的、不确定的、时变的环境中,通过自我学习不断提高蚂蚁的适应性。

但是,基本蚁群算法存在以下缺点:

(1) 与遗传算法等相比,该算法一般需要较长的搜索时间。这是因为蚁群中个体的移动是随机的,虽然通过信息的交流能够向着最优路径进化,但是当问题规模较大时,很难在较短时间内从杂乱无章的路径中找出一条较好的路径,而解的构造过程也会占用大量的计算时间。这一缺点是蚁群算法本身决定的,很难有本质上的改进,但可通过采用局部搜索等方法提高算法收敛性,减少算法搜索到满意解的时间。

(2) 容易出现停滞现象。停滞现象是指当算法搜索到一定程度后,所有蚂蚁不能构造新的解,以致不能对搜索空间做进一步探索的现象。蚂蚁总是倾向沿着信息素强度高的弧段移动,由信息素更新规则可见,未被选取的弧段与包含在优解中的弧段相比,信息素强度的差异越来越大,它们被选择的概率也就越来越小,从而导致算法有时只能在信息素更新中的优解附近进行搜索。这种信息素更新规则实现了正反馈机制,但是停滞现象是这种方式要避免的一个不足之处。所以在算法的求解过程中,需要折中考虑算法的探索(exploration)和开发(exploitation)能力。

(3) 有些优化问题难以用构造图描述。虽然构造图在一定程度上扩展了蚁群算法的应用范围,但许多较复杂的实际问题仍然难以用构造图描述。

2.3.2 蚁群系统

蚁群系统是 Dorigo 和 Gambardella 于 1997 年提出的,蚁群系统与蚂蚁系统主要有以下不同之处。

1) 状态转移规则

在 ACS 中,状态转移规则如下:一只位于结点 i 的蚂蚁按照式(2-7)给出的规则选取下一个将要到达的城市 j,

$$j=\begin{cases}\operatorname{argmax}_{u\in J(i)}\{\tau(i,u)^{\alpha}\eta(i,u)^{\beta}\}, & q\leqslant q_0\\ S, & \text{其他}\end{cases} \tag{2-7}$$

其中，q 是一个$[0,1]$之间的服从均匀分布的随机数，q_0 是一个参数($q_0\in[0,1]$)，S 是按照式(2-1)给出的概率分布选出的一个随机变量，$J(i)$是可选城市集合。

式(2-7)给出的状态转移规则称为伪随机比例状态转移规则(pseudo random-proportional state transition rule)。这个状态转移规则，与式(2-1)给出的随机状态转移规则(random-proportional state transition rule)一样，都倾向于选择较短的且有较多信息素的边作为移动方向。参数 q_0 决定了探索和开发的相对重要性：当一只位于结点 i 的蚂蚁按照式(2-7)给出的规则选取下一个将要到达的城市 j 时，它先产生一个随机数 $0\leqslant q\leqslant 1$；如果 $q\leqslant q_0$，依据式(2-7)选取最好边，否则按照式(2-1)选取一条边。

2) 全局信息素更新规则

在 ACS 中，只有全局最优的蚂蚁才被允许释放信息素。这种策略以及伪随机比例规则的使用，使得蚂蚁具有更强的开发能力：蚂蚁的搜索主要集中到当前迭代为止时所找出的最优路径的邻域内。在每只蚂蚁都构造完一个解之后，全局信息素更新规则按照下式执行：

$$\tau(i,j)\leftarrow(1-\omega)\tau(i,j)+\omega\Delta\tau(i,j) \tag{2-8}$$

$$\Delta\tau(i,j)=\begin{cases}(L_{gb})^{-1}, & (i,j)\in\text{全局最优路径}\\ 0, & \text{其他}\end{cases} \tag{2-9}$$

其中，ω 是信息素挥发参数($0<\omega<1$)，L_{gb} 是全局最优路径长度。

由式(2-8)和式(2-9)可知，只有那些属于全局最优路径的弧段上的信息素才得到增强。全局更新规则的另一个类型是用当前迭代的最优解更新信息素。在这种策略中，式(2-9)中用当前迭代的最优路径长度 L_{ib} 代替 L_{gb}，并且只有属于当前迭代的最优路径中的弧段才会得到增强。实验表明，这两种类型对蚁群系统性能影响的差别很小，全局最优的性能稍好一些。

3) 局部信息素更新规则

在构造解时，蚂蚁应用式(2-10)给出的局部更新规则对它们所经过的边更新信息素：

$$\tau(i,j)\leftarrow(1-r)\tau(i,j)+r\Delta\tau(i,j) \tag{2-10}$$

其中，$r(0<r<1)$是一个参数。

实验表明，当 $\Delta\tau(i,j)=\tau_0$ 时(其中 τ_0 为一常数)，算法能在较短的时间内获得较好的解。特别地，在 TSP 问题中，$\tau_0=(nL_{nn})^{-1}$，其中 L_{nn} 是由最近

邻域启发算法求得路径的长度，n 是城市的数目。

4）蚁群系统采用候选表策略

蚁群系统是一种构造式随机算法，在每一步，如果蚂蚁在选择下一个城市时考虑所有可选的城市，状态转移规则的计算量会很大。为了提高蚁群系统的搜索效率，特别是对于较大规模的问题，蚁群系统采用候选表策略。候选表是一个表，它记录从当前城市出发偏好程度较高的城市（preferred cities）。只要候选表中还有未访问过的城市，蚂蚁就会按照状态转移规则从候选表中选取一个城市。当候选表中的所有城市都被访问过时，蚂蚁才会考虑不在候选表中的城市。

2.3.3 最大最小蚂蚁系统

Stützle 和 Hoos 在 2000 年提出 MMAS。它与 AS 的差异主要体现在信息素更新规则上。

1）MMAS 采用精英策略来更新信息素

具体而言，在每个蚂蚁构造完一个解之后，只增加最优解对应弧段的信息素。这个解可能是当前最优解（best-so-far solution），也可能是当前迭代的最优解（iteration-best solution）。当只使用当前最优解时，搜索可能会过快地集中到这个解的周围，从而限制了对新解的搜索，甚至可能会陷于局部最优解。而利用当前迭代的最优解来更新信息素可以减少这样的风险。这是因为当前迭代的最优解在每次迭代时都可能不同。一般地，利用当前最优解来更新信息素，可使蚁群获得较强的开发能力，而利用当前迭代的最优解来更新信息素，可使蚁群获得较强的探索能力。实验表明，采用混合策略且在迭代过程中增加使用当前最优解进行信息素更新的频率，能有效平衡探索和开发能力，从而提高算法性能。

2）信息素取值限制在区间$[\tau_{\min},\tau_{\max}]$，其中 $\tau_{\min}$、$\tau_{\max}$分别是信息素下界和上界

MMAS 通过将超出这个范围的值强制设置为 $\tau_{\min}$或 $\tau_{\max}$，避免不同弧段的信息强度差异过大，从而达到避免停滞的目的。

3）MMAS 将信息素初始化为 $\tau_{\max}$

在 MMAS 中，信息素的值在第一次迭代之后都被设置为 $\tau_{\max}(1)$。这一点可以通过将信息素的初始值设置为某个非常大的数值来实现。这种策略使得蚂蚁在算法的初始阶段能够具有更好的探索能力。实验表明，它能改善算法的性能。

4）MMAS 还利用信息素的平滑机制提高其性能

当 MMAS 已经收敛或接近收敛时，这种机制将信息素作如下调整：

$$\tau(i,j)^* \leftarrow \tau(i,j) + \delta(\tau_{\max} - \tau(i,j)) \tag{2-11}$$

其中，$0<\delta<1$；$\tau(i,j)$ 与 $\tau(i,j)^*$ 是信息素调整前后的信息素值。信息素平滑机制的基本思想是通过增加选择信息素值较小的解元素的概率提高搜索新解的能力。由于 $\delta<1$，它能避免完全丢失在算法运行过程中所积累的信息。当 $\delta=1$ 时，它相当于信息素的重新初始化。而当 $\delta=0$ 时，该机制不发挥作用。平滑机制有助于改善算法的探索能力。同时，这个机制可以降低 MMAS 对信息素下限的敏感程度，有利于在全局范围内搜索新的解，同时避免过早收敛于局部优解。

2.4 蚁群算法研究现状

自 1991 年 Dorigo 等提出蚁群算法以来[10]，特别是 1996 年 Dorigo[2] 等系统阐述了蚁群算法的基本原理和数学模型之后，蚁群算法倍受关注，取得了大量的应用与理论研究成果。

2.4.1 蚁群算法的应用

在蚁群算法最初发展阶段，它主要应用于解决旅行商问题（TSP）、二次指派问题（QAP）以及作业调度问题（JSP）等为数不多的几类问题。现在已经在图着色问题[11]、车辆路径问题[12]、项目调度[13]、约束满足问题[14]、点覆盖问题[15]、网格划分问题[16]、背包问题[17]等许多复杂静态组合优化上取得了令人瞩目的成果。

蚁群算法在动态组合优化问题中一个最成功的应用是网络路由问题[18~20]。这主要是由于这类问题的内部信息和分布计算、随机动态以及异步网络状态更新等特征与蚁群算法的特征匹配得很好。

蚁群算法还用来解决连续优化问题。Bilchev 和 Parmee[21] 提出了第一个求解连续优化问题的蚁群算法。它由全局搜索与局部搜索两部分组成，利用蚁群算法进行局部搜索，而全局搜索由遗传算法完成。其中，全局搜索确定整个搜索空间的区域，并给这些区域做出简单的评价。新区域的创建通过类似于遗传算法的交叉和变异操作完成。蚂蚁在经过的区域上根据该区域的目标函数值留下信息素，这些信息素可以指导其他蚂蚁进行局部搜索。Dreo 和 Siarry[22] 在该算法的基础上引入了密度等级结构（Dense heterarchy），将信息素间接通信和直接通信两种方式结合起来。陈崚等[23]

提出,将解空间划分成若干子域,根据信息量求出解所在的子域,然后在该子域内已有的解中确定解的具体值。杨勇等[24]提出一种求解连续空间优化问题的蚁群算法,该算法主要包括全局搜索、局部搜索和信息素强度更新规则。在全局搜索过程中,蚂蚁的移动方向由信息素强度和启发式函数确定。在局部搜索过程中,采用确定性搜索改善算法寻优性能,加快收敛速率。李艳君和吴铁军[25]用二进制对每个连续变量编码,然后让蚂蚁在离散域搜索。汪镭等[26]将蚁群算法引入连续空间的函数寻优问题,通过将传统蚁群算法中的"信息量留存"过程拓展为连续空间中的"信息量分布函数",导出了相应的求解算法。程志刚等[27]保留了连续问题可行解的原有形式,并融合演化算法的种群与操作功能。他们将蚁群分为全局和局部蚂蚁,个体分别执行全局探索式和局部挖掘式搜索,并释放信息素,实现信息共享,利用正反馈机制以加速寻优进程。寇晓丽和刘三阳[28]将解空间按一定原则分解成离散子空间,蚂蚁在离散子空间搜索;并且用改进的 Alopex 算法对蚂蚁搜索得到的解进行修正。Socha 和 Dorigo[29]提出了离散域中蚁群算法的一个直接推广,其基本思想是利用概率密度函数(probability density function)替代离散域中的离散概率分布,蚂蚁依据概率密度函数进行抽样。段海滨等[30]提出了一种用于求解连续空间优化问题的改进蚁群算法,将连续空间优化问题的解向量分解成有限个网格,同时构造了一个与蚁群转移概率相关的评价函数,并借助相遇搜索策略对蚁群算法进行了改进,将信息素数量限制在一个有界区间,以提高改进蚁群算法的全局收敛性能。

蚁群算法还广泛用于求解多目标优化问题。张勇德和黄莎白[31]提出了基于蚁群算法的连续多目标算法。他们定义了连续空间中信息量的留存方式和蚂蚁的行走策略,并将信息素交流和基于全局最优经验指导两种寻优方式结合,以加速算法收敛和维持多样性。Doerner 等[32]提出一类基于 pareto 占优的多目标蚁群算法,并用于项目证券投资组合(project portfolio selection)。蚁群算法已经在许多工程中的多目标优化问题得到应用,如多 QoS 约束海量数据网格任务调度[33]、热轧带钢轧制批量计划优化[34]、柔性约束电网规划[35]等。此外,蚁群算法还用来解决复杂混合变量等优化问题[36]。

2.4.2 蚁群算法的改进

针对基本蚁群算法的不足,国内外学者深入研究了信息素释放方式、信息素更新规则、路径选择策略、参数的选取以及并行实现和计算效率等,并

且融合其他算法改进基本蚁群算法。

1）信息素释放方式

当AS应用到TSP问题中，蚂蚁采用边(edge)模式释放信息素，即将信息素释放到边上。随后研究者提出的信息素释放方式还有点(vertex)模式和团(clique)模式[37]等。与这几种方式相对应，信息素分别表征边、点和团的偏好。实验表明，针对不同问题，合理地选取信息素释放方式对于算法性能有着重要影响。

2）信息素更新规则

信息素更新规则是蚁群算法的核心。在AS中更新规则对所有蚂蚁经过的路径上的信息素都进行更新，这就降低了选择最优路径的概率，使得蚂蚁不能很快集中在最优路径的邻域内搜索。为克服这一缺陷，Dorigo和Gambardella[5]提出了全局更新规则和局部更新规则。一方面，由于全局更新规则只对全局最优路径进行信息素的增强，其他路径的信息素由于挥发机制会逐渐减少，这就增大了最优路径与其他路径在信息素上的差异，使得蚂蚁更倾向于选择最优路径中的边，并且很快集中在最优路径的邻域内搜索，从而提高算法的搜索效率。另一方面，局部更新规则使得蚂蚁具有更好的探索新解的能力。在此基础上，Stützle和Hoos[7]提出仅允许最优蚂蚁更新信息素，从而强化算法开发能力，达到提高算法获得更好解的效率的目的。但是这种精英策略(elitistst strategy)往往会导致算法在搜索早期收敛于局部最优解而过早停滞。为此，他们提出限制信息素上下界的方法，并通过把信息素初始化为信息素的上界使算法在初始阶段具有较好的探索能力。

此外，Bullnheimer等[6]将排序的概念扩展到蚂蚁系统，提出了基于排序的蚂蚁系统。该算法在每一次迭代后，将蚂蚁所经路径的长度按从小到大的顺序排列，并根据路径长度赋予不同的权重，路径较短的权重较大。鉴于蚂蚁系统搜索效率低和质量差的缺点，李士勇[8]提出了利用最优最差蚂蚁更新信息素。覃刚力和杨家本[38]采用锦标赛竞争策略更新信息素。陈峻等[39]提出了一种分布均匀度的自适应蚁群算法，根据优化过程中解的分布均匀度，自适应地调整路径选择概率的制定策略和信息素更新策略，以求在加速收敛和防止早熟、停滞现象之间取得很好的平衡。他们[40]还提出了一种具有感觉和知觉特征的蚁群算法，使蚂蚁受显意识和潜意识的相互作用选择路径，同时自适应地修改路径上的信息量。黄国锐等[41]提出了一种基于信息素扩散的蚁群算法，通过建立信息素扩散模型，使相距较近的蚂蚁之间能更好地进行协作。曹先彬和尹宝勇[42]采用异步更新规则调整各个

蚂蚁的信息素强度，从而间接改变蚂蚁间合作方式。

3）路径选择策略

路径选择策略直接影响到解的质量。Dorigo 和 Gambardella[5]在随机比例状态转移规则的基础上，提出了伪随机比例状态转移规则（Pseudo-random-proportional state transition rule）。它提供了一种直接的方法平衡新路径的探索和先验知识以及问题积累知识的开发。郝晋等[43]提出了一种随机扰动蚁群算法，该算法具有一定的自适应性以及很强的随机扰动特性。李万庆和李彦苍[44]提出了基于信息熵的路径选择策略。王一清等[45]利用 Bayes 决策的后验分析技术，改进了基本蚁群算法中的随机搜索策略。

4）参数的选取

蚁群算法的参数是影响其解的质量和计算效率的关键因素。目前尚没有完善的理论指导，一般采用试凑法（trial and error）选定，该方法显然会影响算法的应用和性能。Botee 等[46]用遗传算法求得参数的最优组合，但是该方法比较烦琐。Zecchin 等[47]针对给排水系统优化问题用敏感度分析法给出了参数选取原则。冯远静[48]指出随着运行代数的增加逐渐减小信息素的挥发率，能改进算法性能。段海滨和王道波[49]在分析蚁群算法的全局收敛性的基础上，提出信息素强度变参数控制、挥发系数动态自适应调节策略。邢桂华和于盛林[50]对参数和选择策略进行了分阶段设计，而且参数的分阶段是根据寻优状态动态划分的。段海滨[9]提出用三步走的方法确定参数。Birattari 等[51]采用机器学习的方法设置参数。Pellegrini 等[52]分析了蚂蚁数、挥发率、信息素和启发信息的指数对于 MMAS 收敛速度的影响，提出了选取参数的方法。

5）与其他算法的融合

局部搜索常用来改善蚁群算法的性能[53]。蚁群算法执行一种粗搜索，它为局部搜索提供好的初始解；反过来，局部搜索这种精搜索又有助于避免蚁群算法陷于局部最优解。

利用蚁群算法良好的耦合能力，将蚁群算法和遗传算法相结合是蚁群算法改进的另一途径[54]。蚁群算法具有很强的正反馈能力，在算法后期能够加快算法的进化速度，促使算法迅速收敛，但在算法初期信息素匮乏，进化速度较慢；遗传算法具有快速随机的全局搜索能力，但不能利用系统的反馈信息。采用遗传算法生成信息素初始分布，利用蚁群算法求精确解，能够实现两种算法的优势互补。为了克服蚁群算法求解时间较长的缺陷，吴庆洪等[55]提出了一种具有变异特征的蚁群算法。陈烨[56]引入遗传算法中

的杂交算子。孙焘等[57]提出了一种简单蚁群算法，利用变异算子改善蚁群初始解。段海滨等[58]采用云模型定性关联规则和自然界的小生境思想改善蚁群算法性能。文献[59,60]将蚁群算法和神经网络相结合，利用神经网络广泛映射能力和蚁群算法快速全局收敛能力，可在一定程度上避免神经网络易陷入局部极小点的缺陷。侯云鹤等[61]将粒子群算法应用到蚁群算法的局部搜索中，提高了优化效率和计算精度。文献[62,63]融合免疫系统的适应能力和蚁群算法的全局搜索能力，提出免疫蚁群融合算法，取得了较好的实验效果。

6）并行实现和计算效率

并行实现是改善蚁群算法计算效率的有效途径。Randall 和 Lewis[64]研究了5种并行实现方法：①并行独立蚁群法：一组蚁群算法在处理机上串行实现，每个蚁群的参数可能不同，它的优点是蚁群不需要通信；②并行交互蚁群法：它与并行独立蚁群法的区别在于蚁群间有交互信息；③并行蚂蚁法，它给每个蚂蚁分配一个从处理机，主处理机负责接受用户的输入，确定蚂蚁的初始位置、全局信息素更新、完成输出等任务；④解元素的并行评价法：它在从处理机上评价解元素；⑤混合法：它结合了并行蚂蚁法和解元素的并行评价法。

陈崚和章春芳[65]提出了并行蚁群算法中处理机间信息交流的两种策略，使得各处理机能够自适应地选择其他处理机以进行信息交换和相应信息素的全局更新。他们还提出了一种确定处理机之间进行信息交流的时间的策略，可以根据解的分布情况自适应地确定处理机进行信息交流的时间，以平衡全局收敛速度和解的多样性。在算法每一次信息交换后，采用自适应的更新策略，根据信息素的均匀度进行信息素的更新，以达到避免早熟和局部收敛的目的。

最近，熊伟清和魏平[66]提出了一种二进制蚁群算法。这种算法采用二进制编码。由于这种算法对单个蚂蚁的智能行为要求比较低，对应的存储空间相对较少，从而提高了算法的计算效率。

2.4.3　蚁群算法的理论研究

虽然蚁群算法在应用上取得了丰硕成果，但蚁群算法的理论研究还主要集中于算法的收敛性和随机模型等方面。

Gutjahr[67]用 Markov 过程描述了一类蚁群算法 GBAS，并且证明了其收敛性。Stützle 和 Dorigo[68]证明了一类蚁群算法 $ACO_{gb,\tau_{min}}$ 的收敛性。在此基础上，证明了两类具有代表性的蚁群算法即蚁群系统和最大最小蚂蚁

系统,在给定信息素最小值(信息素下界)τ_{min}和最大值(信息素上界)τ_{max}的条件下以概率1收敛到最优解。Gutjahr[69]还在GBAS的基础上提出了两类新的算法GBAS/tdev和GBAS/tdlb,并证明可以通过选取合适的参数保证算法的收敛性。

Hou等[70]用不动点理论分析了一类广义蚁群算法的收敛性。Yoo等[71,72]分析了一类分布式蚂蚁路由算法的收敛性。Badr和Fahmy[73]用分支随机过程描述蚁群算法,从分支随机路径和分支WIENER过程的角度研究了蚂蚁路径存亡的比率,并证明了该过程为稳态分布。孙焘等[74]提出了一种简单蚁群算法,利用变异算子改善蚁群初始解,并给出了收敛性证明。丁建立等[75]依据Markov理论对一种遗传蚁群融合算法的收敛性进行了分析,证明了其优化解满足值序列单调非增且收敛。

段海滨等[76]以Markov链和离散鞅作为研究工具,对基本蚁群算法收敛性问题进行了理论证明,把最优解集序列转变为下鞅序列考查残留信息素轨迹向量的收敛性,并且给出了基本蚁群算法首达时间的定义,从理论上分析了基本蚁群算法首次到达时间的期望值。

最近,黄翰等[77]基于吸收态Markov过程模型,以期望收敛时间作为研究指标对蚁群算法收敛速度进行了理论分析;根据吸收态的性质给出了期望收敛时间的估算方法,理论上衡量了蚁群算法收敛速度,并且提出了ACO-易和ACO-难问题的判定方法。他们还给出了蚁群算法参数的优化设计理论方法,以确保算法能在多项式时间内求解ACO-易问题。并且针对蚁群系统,分析了算法参数与迭代时间的关系,提出了蚁群系统易求解问题的具体判别规则。

值得指出的是,Blum和Dorigo[78]研究了蚁群算法的搜索偏向(search bias)。针对简化的蚂蚁系统,研究了第一、第二类欺骗问题,分析了竞争平衡系统(competition balance system)与第二类欺骗的关系。他们发现对于竞争平衡系统,可能不会出现第二类欺骗;而对于非竞争平衡系统,可能会发生第二类欺骗。

2.5 小结

本章讲述了蚁群智能优化方法的生物学思想起源,给出了算法的基本框架,并且给出了3个蚁群算法代表。最后,从算法应用、改进与理论研究3个方面总结了国内外重要的研究成果。

参考文献

[1] Goss S, Aron S, Deneubourg J L, et al. Self-organized shortcuts in the Argentine ant[J]. Naturwissenschaften, 1989, 76:579～561.
[2] Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperating agents[J]. IEEE Transactions on System Man, and Cybernetics: Part B, 1996, 26: 29～41.
[3] Dorigo M, Caro G Di, Gambardella L M. Ant algorithms for discrete optimization [J]. Artificial Life, 1999, 5(2): 137～172.
[4] Dorigo M, Caro G Di. The ant colony optimization metaheuristic[C]. In: Corne D, Dorigo M, Glover F, Ed. New Ideas in Optimization, London, U K: McGraw-Hill, 1999: 11～32.
[5] Dorigo M, Gambardella L M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem [J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53～66.
[6] Bullnheimer B, Hartl R F, Strauss C. A new rank based version of the ant system: A computational study[J]. Central European Journal for Operations Research and Economics, 1999, 7(1): 25～38.
[7] Stützle T, Hoos H H. MAX-MIN ant system[J]. Future Generation Computer Systems, 2000, 16(8): 889～914.
[8] 李士勇. 蚁群算法及其应用[M]. 哈尔滨：哈尔滨工业大学出版社，2004.
[9] 段海滨. 蚁群算法原理及其应用[M]. 北京：科学出版社，2005.
[10] Colorni A, Dorigo M, Maniezzo V, et al. Distributed optimization by ant colonies [C]. Proceedings of the 1st European Conference on Artificial Life, 1991: 134～142.
[11] Costa D, Hertz A. Ants can colour graphs[J]. Journal of the Operational Research Society, 1997, 48: 295～305.
[12] Gambardella LM, Taillard ED, Agazzi G. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows[C]. Corne D, Dorigo M, Glover F, Ed. New Ideas in Optimization, London, U K: McGraw-Hill, 1999: 63～76.
[13] Merkle D, Middendorf M, Schmeck H. Ant colony optimization for resource constrained project scheduling [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 333～346.
[14] Solnon C. Ants can solve constraint satisfaction problems[J]. IEEE Transactions on Evolutionary Computation, 2002, 6(4): 347～357.
[15] Lessing L, Dumitrescu I, Stützle T. A comparison between ACO algorithms for the set covering problem[J]. Lecture Notes in Computer Science, 2004, 3172: 1～12.

[16] Korosec P，Silc J，Robic B. Solving the mesh-partitioning problem with an ant-colony algorithm[J]. Parallel Computing，2004，30：785～801.
[17] 秦玲，白云，章春芳，等. 解 0-1 背包问题的蚁群算法[J]. 计算机工程，2006，32(6)：212～214.
[18] 王颖，谢剑英. 一种基于蚁群算法的多媒体网络多播路由算法[J]. 上海交通大学学报，2002，36(4)：526～529.
[19] 陈岩，杨华江，朱华勇，等. 基于树分解/合并策略的 QoS 多播路由方法[J]. 国防科技大学学报，2007，29(2)：117～122.
[20] 秦玲，陈崚，周日贵，等. 一种基于蚁群系统的组播路由算法[J]. 信息与控制，2006，35(5)：549～550.
[21] Bilchev G，Parmee I C. The ant colony metaphor for searching continuous design spaces[J]，Lecture Notes in Computer Science，1995，993：25～39.
[22] Dreo J，Siarry P. Continuous interacting ant colony algorithm based on dense heterarchy[J]. Future generation computer systems，2004，20：841～856.
[23] 陈崚，沈洁，秦玲. 蚁群算法求解连续空间优化问题的一种方法[J]. 软件学报，2002，13(12)：2317～2323.
[24] 杨勇，宋晓峰，王建飞，等. 蚁群算法求解连续空间优化问题[J]. 控制与决策，2003，18(5)：573～576.
[25] Li Y J，Wu T J. An adaptive ant colony system algorithm for continuous space optimization problems[J]. Journal of Zhejiang University-SCIENCE，2003，4(1)：40～46.
[26] 汪镭，吴启迪. 蚁群算法在系统辨识中的应用[J]. 自动化学报，2003，29(1)：102～109.
[27] 程志刚，陈德钊，吴晓华. 连续蚁群优化算法的研究[J]. 浙江大学学报，2005，39(8)：1147～1151.
[28] 寇晓丽，刘三阳. 一种求解连续优化的蚁群混合算法[J]. 西安电子科技大学学报，2006，33(5)：745～747.
[29] Socha K，Dorigo M. Ant colony optimization for continuous domains[J]. European Journal of Operational Research，2007.
[30] 段海滨，马冠军，王道波，等. 一种求解连续空间优化问题的改进蚁群算法[J]. 系统仿真学报，2007，19(5)：974～977.
[31] 张勇德，黄莎白. 多目标优化问题的蚁群算法研究[J]. 控制与决策，2005，20(2)：172～176.
[32] Doerner K，Gutjahr W J，Hartl R F，et al. Pareto ant colony optimization with ILP preprocessing in multiobjective project portfolio selection[J]. European Journal of Operational Research，2006，171：830～841.
[33] 胡毅，龚斌，刘运臣. 基于蚁群算法的多 QoS 约束海量数据网格任务调度[J]. 华中科技大学学报，2007，35：90～93.
[34] 刘士新，宋健海，周山长. 热轧带钢轧制批量计划优化模型及算法[J]. 控制理论与应用，2007，24(2)：243～248.

[35] 翟海保，程浩忠，陈春霖，等. 基于最小期望投资悔值的柔性约束电网灵活规划方法[J]. 上海交通大学学报，2005，39：27～31.

[36] 闻育，吴铁军. 求解复杂多阶段决策问题的动态窗口蚁群优化算法[J]. 自动化学报，2004，30(6)：872～879.

[37] Fenet S，Solnon C. Searching for maximum cliques with ant colony optimization [J]. Lecture Notes in Computer Science，2003，26(11)：291～302.

[38] 覃刚力，杨家本. 自适应调整信息素的蚁群算法[J]. 信息与控制，2002，31(3)：198～201.

[39] 陈崚，沈洁，秦玲，等. 基于均匀分布度的自适应蚁群算法[J]. 软件学报，2003，14(8)：1379～1387.

[40] 陈崚，沈洁，秦玲，等. 具有感觉和知觉特征的蚁群算法[J]. 系统仿真学报，2003，15(10)：1418～1425.

[41] 黄国锐，曹先彬，王煦法. 基于信息素扩散的蚁群算法[J]. 电子学报，2004，32(5)：865～868.

[42] 曹先彬，尹宝勇. 基于信息素异步更新的蚁群算法[J]. 系统工程与电子技术，2004，26(11)：1680～1683.

[43] 郝晋，石立宝，周家启. 具有随机扰动特性的蚁群算法[J]. 仪器仪表学报，2001，22(4)：349～352.

[44] 李万庆，李彦苍. 求解复杂优化问题的基于信息熵的自适应蚁群算法[J]. 数学的实践与认识，2005，35(2)：134～139.

[45] 王一清，宋爱国，黄惟一. 基于 Bayes 决策的蚁群优化算法[J]. 东南大学学报，2005，35(4)：558～562.

[46] Botee H M，Bonabeau E. Evolutionary ant colony optimization[J]. Advances in Complex Systems，1998，1(2)：149～159.

[47] Zecchin A C，Simpson A R，Marier H R，et al. Parametric study for an ant algorithm applied to water distribution system optimization [J]. IEEE Transactions on Evolutionary Computation，2005，9(2)：175～191.

[48] 冯远静. 群体协同蚁群算法及其在图像分割中的应用[D]. 西安：西安交通大学，2004.

[49] 段海滨，王道波. 蚁群算法的全局收敛性研究及改进[J]. 系统工程与电子技术，2004，26(10)：1506～1509.

[50] 邢桂华，于盛林. 动态分阶段蚁群算法及其收敛性分析[J]. 控制与决策，2007，22(6)：685～688.

[51] Birattari M，Stiitzle T，Paquete L，et al. A racing algorithm for configuring metaheuristics[C]. Langdon W B，et al. ed. Proceedings of the Genetic and Evolutionary Computation Conference（GECCO-2002），CA：Morgan Kaufmann Publishers，2002：11～18.

[52] Pellegrini P，Favaretto D，Moretti E. On MAX—MIN ant system's parameters [C]. Dorigo M. et al. ed. ANTS 2006，Lecture Notes in Computer Science，2006，41(50)：203～214.

[53] Dorigo M, Stützle T. The ant colony optimization metaheuristic: algorithms, applications, and advances[C]. Glover F, Kochenberger G. Ed. Handbook of Metaheuristics. MA: Kluwer Academic Publishers, 2002: 251～285.
[54] 丁建立,陈增强,袁著祉. 遗传算法与蚂蚁算法融合的马尔可夫收敛性分析[J]. 自动化学报, 2004, 30(4): 629～634.
[55] 吴庆洪,张纪会,徐心和. 具有变异特征的蚁群算法[J]. 计算机研究与发展, 1999, 36(10): 1239～1245.
[56] 陈烨. 带杂交算子的蚁群优化算法[J]. 计算机工程, 2001, 27(12): 74～76.
[57] 孙焘,王秀坤,刘业欣,等. 一种简单蚂蚁算法及其收敛性分析[J]. 小型微型计算机系统, 2003, 24(8): 1524～1527.
[58] 段海滨,王道波,于秀芬. 基于云模型的小生境 MAX-MIN 相遇蚁群算法[J]. 吉林大学学报, 2006, 36(5): 803～808.
[59] 洪炳熔,金飞虎,高庆吉. 基于蚁群算法的多层前馈神经网络[J]. 哈尔滨工业大学学报,2003, 35(7): 823～825.
[60] 邹政达,孙雅明,张智晟. 基于蚁群优化算法递归神经网络的短期负荷预测[J]. 电网技术,2005, 29(3): 59～63.
[61] 侯云鹤,鲁丽娟,熊信艮,等. 广义蚁群与粒子群结合算法的电力系统经济负荷分配[J]. 电网技术, 2004, 28(21): 34～38.
[62] Feng Y, Feng Z. An immunity-based ant system for continuous space multi-modal function optimization[C]. Proceedings of the third international conference on Machine Learning and Cybernetics, 2004: 1050～1054.
[63] 蒋加伏,陈荣元,唐贤瑛,等. 基于免疫-蚂蚁算法的多约束 QoS 路由选择[J]. 通信学报, 2004, 25(8): 89～95.
[64] Randall M, Lewis A. A parallel implementation of ant colony algorithm[J]. Journal of Parallel and Distributed Computing, 2002, 62: 1421～1432.
[65] Chen L, Zhang C F. Adaptive exchanging strategies in parallel ant colony algorithm[J]. Journal of software, 2007, 18(3): 617～624.
[66] 熊伟清,魏平. 二进制蚁群进化算法[J]. 自动化学报, 2007, 33(3): 259～264.
[67] Gutjahr W J. A graph-based ant system and its convergence[J]. Future Generation Computer Systems, 2000, 16(8): 873～888.
[68] Stützle T, Dorigo M. A short convergence proof for a class of ant colony optimization algorithm[J]. IEEE Transactions on Evolutionary Computation. 2002, 6(4): 358～365.
[69] Gutjahr W J. ACO algorithms with guaranteed convergence to the optimal solution[J]. Information Processing Letters, 2002, 82(3): 145～153.
[70] Hou Y H, Wu Y W, Lu L J, et al. Generalized ant colony optimization for economic dispatch of power systems[C]. Proceedings of the 2002 International Conference on Power System Technology, 2002, 1: 225～229.
[71] Yoo J H, La R J, Makowski A M. Convergence results for ant routing[R]. Technical Report CSHCN 2003-46, Institute for Systems Research, University of

Maryland，College Park(MD)，2003.

［72］ Yoo J H，La R J，Makowski A M. Convergence of ant routing-results for simple parallel network and perspectives［R］. Technical Report CSHCN 2003-44，Institute for Systems Research，University of Maryland，College Park (MD)，2003.

［73］ Badr A，Fahmy A. A proof of convergence for ant algorithms［J］. International Journal of Intelligent Computing and Information，2003，3(1)：22～32.

［74］ 孙焘，王秀坤，刘业欣，等. 一种简单蚂蚁算法及其收敛性分析［J］. 小型微型计算机系统，2003，24(8)：1524～1527.

［75］ 丁建立，陈增强，袁著祉. 遗传算法与蚂蚁算法融合的马尔可夫收敛性分析［J］. 自动化学报，2004，30(4)：629～634.

［76］ 段海滨，王道波，于秀芬. 基本蚁群算法的 A. S. 收敛性研究［J］. 应用基础与工程科学学报，2006，14(2)：297～301.

［77］ 黄翰，郝志峰，吴春国，等. 蚁群算法的收敛速度分析［J］. 计算机学报，2007，30(8)：1344～1353.

［78］ Blum C，Dorigo M. Search bias in ant colony optimization：on the role of competition-balanced systems［J］. IEEE Transactions on Evolutionary Computation，2005，9(2)：159～174.



旅行商问题

3.1 引言

旅行商(TSP)问题是一类经典优化问题,与它相关的优化算法不胜枚举。经典蚁群算法已经用于求解旅行商问题,并得到很好的结果。但是,在蚁群算法中,信息素直接影响着整个搜索过程。在信息素更新时利用到目标函数值,这固然有其合理性,但目标函数值的变化规律难以预知,这给算法的参数设置带来很大的困难。另外,Blum 和 Dorigo 指出对于两个呈常数比例的目标函数,即使基本蚁群算法采用完全相同的参数,算法性能却可能不同,文献[2]指出在一定条件下算法具有不变性。这显然不是人们所期望的[1]。针对这些问题,本章提出一种有限级信息素蚁群算法。

3.2 算法描述

实验发现,MMAS 和 ACS 具有以下共同特点:

(1) 参数的设置是依赖问题的,几乎没有规律可循,而且算法对参数比较敏感。

(2) 在算法的运行过程中,信息素是按照指数下降的,这是算法易于早熟的一个原因。

(3) 在算法的运行过程中,大量的信息素相同或相近。实际上,如果弧上的信息素相近,它们被选取的概率差异非常小,可以近似地把它们看成是等同的。

受此启发,有限级信息素蚁群算法把信息素分成有限个级别,用完全不同的方式更新信息素,并且信息素的更新量与目标函数值无关。为了阐述其工作原理并研究其性能,以 TSP 为测试问题。算法的主要流程如下:

步骤 1,设定参数,初始化信息素;

步骤 2,按照路径选择规则构造问题的解;

步骤 3,按照信息素更新规则更新信息素;

步骤 4,判断停止条件是否满足,若满足,算法终止;否则返回到步骤 2。

在算法实现中,信息素被分成有限个级别,不同的级别按照一定的映射关系对应不同实数值,这样相同级别的弧的信息素相同。信息素更新通过级别的变动实现,对于当前最优弧,提高其级别,对于其他的弧,降低其级别。更新时只用加、减法。记 $h(i,j)$ 是弧 (i,j) 上的级别,$g(x)$ 是单调增的正实函数,它实现从级别到信息素的映射关系,信息素更新规则如下:

(u1) $\forall(i,j)$: $h(i,j) \leftarrow h(i,j)-r_1$;

(u2) 如果 $f(\hat{w})>f(w_t)$,则 $\hat{w}=w_t$;

(u3) 对于 $\hat{w}$, $\forall(i,j)\in\hat{w}$, $h(i,j)\leftarrow h(i,j)+r_2$;

(u4) $\forall(i,j)$: $h(i,j)\leftarrow \max(1,h(i,j))$;

(u5) $\forall(i,j)$: $h(i,j)\leftarrow \min(M,h(i,j))$;

(u6) $\forall(i,j)$: $\tau(i,j)=g(h(i,j))$。

其中,f 为目标函数,$\hat{w}$ 是当前最优解,w_t 是本次迭代的最优解,参数 r_1、r_2 是正整数,$r_1<r_2$,称 r_1 为惩罚级别数,参数 r_1 固定为 1,r_2 为奖励级别数,参数 M 是最大级别数。$\tau_{\max}$ 是信息素上界。约定 $g(M)=\tau_{\max}$,$g(1)=1$。在实验中,$g(x)$ 一般选取为 $\frac{\tau_{\max}-1}{M-1}(x-1)+1$ 或 $\sqrt{\frac{\tau_{\max}^2-1}{M-1}(x-1)+1}$。前一个函数是线性函数,后一个函数是凹函数。这两个函数可以使信息素随着级别下降而衰减速度不变或逐渐变慢。

在信息素更新规则中,通过函数 $g(x)$、奖励级别数、惩罚级别数协调控制级别的变化,进而控制信息素的变化。一方面,使蚁群能有效地在最优解的邻域搜索;另一方面,使蚁群具有一定的探索新区域的能力。由于更新量与目标函数值无关,因此对于两个呈常数比例的目标函数,如果算法采用完全相同的参数,算法性能将相同。

基本蚁群算法用两个参数表征信息素和启发信息的相对重要程度,而在 FGPACO 中把第一个参数内嵌到函数 $g(x)$ 中,相应的路径选择规则如下。

假设蚂蚁在第 k 步位于第 i 个结点，它按照式(3-1)计算选择弧(i,j)的概率：

$$P(c_{k+1}=j \mid c_k=i,\tau)=\begin{cases}\dfrac{\tau(i,j)\cdot\eta(i,j)^{\beta}}{\sum\limits_{r\in J_i}\tau(i,r)\cdot\eta(i,r)^{\beta}}, & j\in J_i\\ 0, & \text{其他}\end{cases} \tag{3-1}$$

其中，$\eta(i,r)$表示弧(i,r)上的启发信息。在 TSP 问题中，启发信息一般选取为弧长的倒数。J_i 是由所有未经过的点组成的集合。

3.3 算法随机模型与收敛性质分析

下面讨论算法的随机模型，分析算法与有限马氏链之间的关系。有关随机过程和有限马氏链的内容见文献[3]。本节中的记号及其意义见表 3-1。马氏链有如下重要性质。

表 3-1 本节用到的记号及其意义

记 号	意 义
$\boldsymbol{\tau}(n)$	第 n 次迭代时的信息素场，它是 $N\times N$ 阶矩阵，N 是问题的规模
$\underline{w}(n)$	第 n 次迭代之前的最优解，($\underline{w}(1)$为任意可行解)，它是 N 维向量
s_n	$s_n=(\boldsymbol{\tau}(n),\underline{w}(n))$，它表示第 n 次迭代时的状态
X	$X=\{s_n\}_{n=1}^{+\infty}$
Γ	s_n 所有可能取值构成的集合，它是有限集
Z	$\underline{w}(n)$所有可能取值构成的集合，它是有限集
$H:\Gamma\rightarrow Z$	H 把状态集 Γ 中的元素映射到 Z 中，$H(\boldsymbol{\tau}(n),\underline{w}(n))=\underline{w}(n)$
$G_{(i,j)}:\Gamma\rightarrow\mathscr{R}$	$\mathscr{R}$ 是实数集，$G_{(i,j)}(s_n)=\boldsymbol{\tau}(n)(i,j)$，$(i,j)\in H(s_n)$，$G_{(i,j)}$ 把状态集映射到实数集
$P_{s_n s_m}$	状态 s_n 转移到状态 s_m 的概率
$\boldsymbol{P}$	状态转移矩阵
π_0	初始状态的概率分布
π_i	第 i 次迭代时的概率分布
π^*	最优状态概率分布

引理 1[3] 对于有限状态马氏链，不管从何状态出发，经过 n 步转移到任一瞬时状态的概率随着 n 无限增大而趋近于 0，反之，离开所有瞬时状态的概率随着 n 无限增大而趋近于 1。

性质 1 在 FGPACO 算法中，X 是有限状态马氏链。

证明：X 是有限状态的；由算法的路径选择规则可知，X 是马氏链。

证毕。

性质 2 在 FGPACO 算法中，若 $s_n, s_m \in \Gamma$，关于 s_n 到 s_m 的转移概率 $P_{s_n s_m}$ 有以下结论：

(1) 若 $f(H(s_n)) < f(H(s_m))$，则 $P_{s_n s_m}=0$。

(2) 若 $H(s_n)=H(s_m)$，且存在某个弧 $(i,j)\in H(s_n)$，使得 $G_{(i,j)}(s_n) > G_{(i,j)}(s_m)$，则 $P_{s_n s_m}=0$。

证明：由信息素更新规则中的(u1)、(u2)、(u3)，并且由定义可以得出结论。

证毕。

这说明状态转移时，或者当前最优解改变，且目标函数值下降；或者当前最优解不变，但它对应的每段弧的信息素递增。如果当前最优解不变，由(u1)、(u3)，每段弧的信息素是收敛的(由单调有界性可知)。经过有限次迭代后，所有弧的信息素将保持不变，此时当前最优解对应的弧上信息素都是 $\tau_{\max}$，其他弧上的信息素都是 1。按照停留时间可以对有限马氏链的状态进行分类。

定义 1(滞留态、最优态和正常态) 对于 $s\in\Gamma$，如果当前最优解对应的弧上信息素都是 $\tau_{\max}$，其他弧上的信息素都是 1，则称 s 是滞留态；特别地，当 $H(s)$ 是全局最优解时，则称 s 为最优态，记为 s^*；其他状态称为正常态。

定理 1(FGPACO 的概率特性) 对于 FGPACO，具有状态 $s_n=(\underline{\boldsymbol{\tau}}(n), \underline{\boldsymbol{w}}(n))$ 的随机过程是有限状态马氏链，它具有以下性质：

(1) 所有状态不是瞬时态，就是吸收态，并且吸收态 s^* 是最优态，它具有正常返性。

(2) 离开正常态只需一次迭代，离开每一个滞留态的迭代次数服从几何分布。

证明：(1) 对于最优态 s^*，由(u1)、(u2)、(u3)可知它只可能转移到自己，即 $\boldsymbol{P}_{s^* s^*}=1$，因此是吸收态，从而具有正常返性。由性质 2 可知其他状态都是瞬时态。

(2) 对于滞留态，记其对应的当前最优解为 w。因为在离开该滞留态之前，每段弧上的信息素保持不变，从而选路概率也不变，因此它服从几何分布。

证毕。

定理 2(FGPACO 的收敛性) FGPACO 算法收敛到全局最优解的概率随着 n 无限增大而趋近于 1。

证明：注意到 FGPACO 每次迭代时，都保留了最优解，由性质 2、引理 1 及定理 1 的结论(1)，可知结论成立。

证毕。

对 Γ 中的元素按照以下约定排序：①对应目标函数值升序排列；②如果目标函数值相同，最优解对应信息素降序排列，其他弧上信息素升序排列。

性质 3 概率转移矩阵 $\boldsymbol{P}$ 是下三角矩阵，且 $\boldsymbol{P}=\begin{bmatrix}\boldsymbol{I}_k & 0\\ \boldsymbol{A} & \boldsymbol{B}\end{bmatrix}$，$\boldsymbol{I}_k$ 是单位矩阵，$\boldsymbol{B}$ 是下三角矩阵，它的最大特征值 λ 小于 1。

证明：注意到 Γ 中的元素按照上面的约定排序，由性质 2 和定理 1 可知结论成立。

证毕。

由于 $\boldsymbol{B}$ 的最大特征值小于 1，则 $(\boldsymbol{I}-\boldsymbol{B})^{-1}$ 是可逆阵。记 $\boldsymbol{M}_i=\boldsymbol{I}+\boldsymbol{B}+\boldsymbol{B}^2+\cdots+\boldsymbol{B}^{i-1}$，则 $\boldsymbol{P}^i=\begin{bmatrix}\boldsymbol{I}_k & 0\\ \boldsymbol{M}_i\boldsymbol{A} & \boldsymbol{B}^i\end{bmatrix}$，$\boldsymbol{P}^\infty=\begin{bmatrix}\boldsymbol{I}_k & 0\\ (\boldsymbol{I}-\boldsymbol{B})^{-1}\boldsymbol{A} & 0\end{bmatrix}$。对于任意矩阵 $\boldsymbol{A}$（行(列)向量可以看成行(列)数为 1 的矩阵），记其范数为 $\|\boldsymbol{A}\|$（有关矩阵范数参见文献[4]）。令 $\boldsymbol{B}$ 的范数为 λ，算法的收敛速度满足以下结论。

定理 3（FGPACO 的收敛速度） π_i 收敛到 π^* 的收敛速度满足 $\|\pi_i-\pi^*\|\leqslant O(\lambda^i)$，也就是说，算法以不大于 λ 的收敛比率收敛。

证明：
$$
\begin{aligned}
\|\pi_i-\pi^*\| &= \|\pi_0\boldsymbol{P}^i-\pi_0\boldsymbol{P}^\infty\|\\
&=\left\|\pi_0\left[\begin{bmatrix}\boldsymbol{I}_k & 0\\ \boldsymbol{M}_i\boldsymbol{A} & \boldsymbol{B}^i\end{bmatrix}-\begin{bmatrix}\boldsymbol{I}_k & 0\\ (\boldsymbol{I}-\boldsymbol{B})^{-1}\boldsymbol{A} & 0\end{bmatrix}\right]\right\|\\
&=\left\|\pi_0\begin{bmatrix}0 & 0\\ (\boldsymbol{M}_i-(\boldsymbol{I}-\boldsymbol{B})^{-1})\boldsymbol{A} & \boldsymbol{B}^i\end{bmatrix}\right\|\\
&=\left\|\pi_0\begin{bmatrix}0 & 0\\ -(\boldsymbol{I}-\boldsymbol{B})^{-1}\boldsymbol{B}^{i+1}\boldsymbol{A} & \boldsymbol{B}^i\end{bmatrix}\right\|\\
&\leqslant \|\pi_0\|\,O(\lambda^i)=O(\lambda^i)
\end{aligned}
$$

证毕。

3.4 参数设置和数值实验分析

本节用 TSPLIB 中的 TSP 算例来测试 FGPACO 算法。沿用 TSPLIB 中的记法，算例中的数字表示城市的数目(算例 ft70 的城市数目是 70，算例 kro124p 是个例外，它的城市数目是 100)。在实现算法时，用一张表记录已经选用的城市，而用另一张表记录未被选用的城市。

3.4.1 参数设置

算法中涉及的参数有信息素的初始值、奖励级别数、最大级别数 M、参数 β 和 $\tau_{\max}$。下面分析参数对算法性能的影响。在实验中，只改变一个参数，而保持其他参数不变。在对算例测试时，$\beta=5$，信息素的初始值取为最大值的 1/2。算法运行 10 次，迭代次数为 2500，蚂蚁数为 25。函数 $g(x)=\frac{\tau_{\max}-1}{M-1}(x-1)+1$。

1）最大级别数 M

表 3-2 给出了 M 取不同值时的实验结果。由表中的结果可见，在 M 等于 10000 时，平均值较大。这时 M 值较大而奖励级别数小导致信息素变化小，算法表现出盲目随机性。因此，在 M 较大时，r_2 应取较大的数。对于算例 eil51，M 取其他 4 个数值时，偏差很小；对于算例 ry48p，M 取值为 50、100、1000 时，偏差很小。结果表明，有限个级别能保证算法收敛到较好的结果，并且算法具有较好的鲁棒性。

表 3-2 $r_2=3$，$\tau_{\max}=200$，最大级别数不同时，平均值及偏差比率

M	10	50	100	1000	10000
ry48p	14571.5 （1.0%）	14534.5 （0.7%）	14545.3 （0.8%）	14544.6 （0.8%）	15758.9 （9.2%）
eil51	429.2 （0.8%）	429.8 （0.9%）	428.1 （0.5%）	428.6 （0.6%）	460.5 （8%）

注：括号内是偏差。

2）信息素上界 $\tau_{\max}$

$\tau_{\max}$ 是影响算法性能的一个重要的量。如果其取值较大，则算法可能停滞；而如果其取值过小，则会减弱算法的开发能力。表 3-3 给出了 $\tau_{\max}$ 不同时的实验结果。由表中结果可见，当 $N\leqslant\tau_{\max}\leqslant 6N$ 时，两个算例的偏差都小于 1%。而当 $\tau_{\max}$ 取值为 10、1000 或者 3000 时，偏差较大。尤其是算例 ry48p 的实验结果，在当 $\tau_{\max}$ 取值为 10 或 3000 时，偏差都大于 2.0%。

3）奖励级别数 r_2

在前面的分析中已经知道 M 较大时，r_2 应取较大的数，反之，M 较小时，r_2 应取较小的数。表 3-4 给出了 r_2 不同时的实验结果。r_2 较小的两个算例结果较好。但算例 eil51 表明 r_2 越小，结果不一定越好。

表 3-3 $r_2=3$,$M=50$,信息素上界不同时,平均值及偏差比率

τ_{max}	10	50	300	1000	3000
ry48p	14728.4 (2.1%)	14491.0 (0.5%)	14547.1 (0.9%)	14646.9 (1.6%)	14713.4 (2.0%)
eil51	431 (1.2%)	428.3 (0.6%)	428.4 (0.6%)	430.1 (1.0%)	429.3 (0.8%)

注:括号内是偏差。

表 3-4 $M=50$,$\tau_{max}=50$,r_2 不同时,平均值及偏差比率

r_2	2	5	10
ry48p	14519.1 (0.7%)	14584.5 (1.1%)	14603.6 (1.2%)
eil51	427.3 (0.3%)	426.9 (0.2%)	427.9 (0.4%)

注:括号内是偏差。

图 3-1 是在信息素上界 $\tau_{max}=50$、最大级别数 $M=50$、奖励级别数 $r_2=3$ 时,算例 ry48p 的实验结果(因为前 500 次迭代已找到较好的解,仅给出前 500 次迭代结果)。由图 3-1 可见,算法收敛速度较快,并且表现出较好的搜索能力。

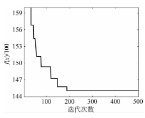


图 3-1 算例 ry48p 求解过程演化

图 3-2 是在信息素上界 $\tau_{max}=50$、最大级别数 $M=50$、奖励级别数 $r_2=3$ 时的算例 eil51 实验结果(同样地,仅给出前 500 次迭代结果)。由图 3-2 可见,“锯齿”较多,这意味着算法能较快地搜索到更好的解。这再一次表明算法具有较好的搜索能力。

3.4.2 与其他改进蚁群算法的比较

依照文献[5]中提议的方式进行测试和比较,每个算例构造 $10000kN$

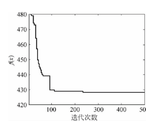


图 3-2 算例 eil51 求解过程演化

次解，对于 TSP，$k=1$，对于 ATSP，$k=2$，N 表示城市数目。奖励级别数 $r_2=3$，最大级别数 $M=50$。下面比较 FGPACO 与 MMAS、MMAS＋pts（具有信息素平滑机制的 MMAS）、ACS、AS 的性能，后 3 类算法的参数设置与数据引自文献[5]。前 4 个算例中 $g(x)=\frac{\tau_{\max}-1}{M-1}(x-1)+1$，在 ft70 和 kro124p 算例中，$g(x)=\sqrt{\frac{\tau_{\max}^2-1}{M-1}(x-1)+1}$。由表 3-5 可知，ft70 算例结果与最优结果很接近，对于其他算例，FGPACO 算法比其他算法的平均结果好。结果表明算法是有效的，且算法不易于早熟收敛。

表 3-5 对称 TSP（前 3 个）和非对称 TSP（后 3 个）的计算结果

算例	最优解	FGPACO ($\tau_{\max}$)	MMAS	MMAS＋pts	ACS	AS
eil51	426	**426.8** (50)	427.6	427.1	428.1	437.3
kroA100	21282	**21286.0** (400)	21320.3	21291.6	21420.0	22471.4
d198	15780	**15950.8** (800)	15972.5	15956.8	16054.0	16702.1
ry48p	14422	**14498.2** (50)	14553.2	14523.4	14565.4	15296.4
ft70	38673	38979.9 (210)	39040.2	**38922.7**	39099.0	39596.3
kro124p	36230	**36444.6** (300)	36773.5	36573.6	36857.0	38733.1

注：粗体字表示最好的平均结果（括号内是 $\tau_{\max}$）。

3.5 小结

基本蚁群算法在信息素更新时利用目标函数值，但目标函数值变化的规律难以预知，这给算法的参数设置带来很大的困难。本章提出的蚁群算法采用了一种新的信息素更新规则，它把信息素分成有限个级别，信息素的更新由级别的更新实现，其主要特点是信息素的离散性以及信息素更新量独立于目标函数值。文中证明了算法的全局收敛性，分析了算法的收敛速度，讨论了算法参数设置的方法。实验证明了算法的有效性、鲁棒性。

虽然本章的研究主要针对 TSP 问题，但是有限级信息素蚁群算法的应用并不局限于这一问题。

参考文献

[1] Blum C, Dorigo M. The hyper-cube framework for ant colony optimization[J]. IEEE Transactions on Systems Man and Cybernetic: Part B, 2004, 34(2): 1161～1172.

[2] Birattari M, Pellegrini P, Dorigo M. On the invariance of ant colony optimization [J]. IEEE Transactions on Evolutionary Computation, 2007, 11(6): 732～742.

[3] Kemeny J G, Snell J L. Finite Markov chains with a new appendix "Generalization of a Fundamental Matrix"[M]. New York: Springer-Verlag, 1976.

[4] 方保镕，周继东，李医民. 矩阵论[M]. 北京：清华大学出版社，2004.

[5] Stützle T, Hoos H H. MAX-MIN ant system[J]. Future Generation Computer Systems, 2000, 16(8): 889～914.



多维背包问题

4.1 问题描述

多维背包问题的优化目标是在满足一定资源(resource)约束条件下从原物品集中选取总利润(profit)最大的物品子集。多维背包问题是一类NP-hard问题,并且许多实际问题可用该问题建模,如货物装载(cargo loading)问题[1]、下料(cutting stock)问题和分布式计算机系统中的处理器和数据库分配(allocating processors and databases in a distributed computer system)问题[2]等。

多维背包问题可以描述为[3]

$$\max \quad \sum_{j=1}^{n} p_j x_j \tag{4-1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} r_{ij} x_j \leqslant b_i, \quad i = 1,2,\cdots,m \tag{4-2}$$

$$x_j \in \{0,1\}, \quad j = 1,2,\cdots,n \tag{4-3}$$

其中,p_j 是物品 j 对应的利润;x_j 是一个二值变量,它标记物品 j 是否被选取,如果 x_j 等于1,这意味着物品 j 被选取;如果 x_j 等于0,就表示物品 j 没有被选取;r_{ij} 是选取物品 j 所花费的第 i 种资源;b_i 是第 i 个资源的总量。在这个问题中有 m 个资源约束,因此该问题又常常被称为 m 维背包问题。

令 $I=\{1,2,\cdots,m\}$，$J=\{1,2,\cdots,n\}$。对于任意 $i\in I$，有 $b_i\geqslant 0$，且对于任意 $i\in I$，$j\in J$，有 $r_{ij}\geqslant 0$。一个定义明确的(well-defined)多维背包问题假设 $p_j>0$，且 $r_{ji}\leqslant b_i<\sum_{j=1}^{n} r_{ij}$。显然地，一个多维背包问题的解 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$是一个 n 维向量。

4.2 现有算法回顾

为了求解多维背包问题，人们提出了许多精确算法(exact algorithm)和启发式算法(heurisitic algorithm)。现有的精确算法主要是基于分支定界法的算法。这些算法的区别在于上界的选取方法。Shih[1]先求解每个约束的一维背包线性松弛问题，再把最小目标函数值作为上界。Gavish 和 Pirkul[4]使用拉格朗日(Lagrangian)松弛法等求得更紧的上界。由于选用了更紧的上界，精确算法获得了更好的计算结果。但是由于计算复杂性，精确算法仅能求解中小规模的问题(例如 $n\leqslant 200, m\leqslant 5$)。

启发式算法已经用于求解更大规模的问题。最初人们考虑贪婪算法。这些算法采用一定的贪婪规则每次增加一个不违反约束的物品。第二类启发式算法是通过求解多维背包问题的线性规划松弛问题来获得近似解。还有一种方法是先松弛整数约束，再用单纯形法求得松弛问题的最优解；其他方法还有共轭综合(surrogate and composite)松弛法等[5]。元启发算法为大规模多维背包问题提供了有效解决方案。研究表明禁忌搜索[6,7]、遗传算法等能提供高质量的解。特别地，Chu 和 Beasley[2]提出的遗传算法以及 Vasquez 和 Hao[8]提出的 z^* 算法在经典算例中得到了很好的解。

最近，文献[9～11]提出了 3 种求解多维背包问题的蚁群算法。在求解多维背包问题时，先将问题建模为一个构造图(construction graph)，图的顶点对应于每个物品，每个边对应于两个顶点(物品)之间的连线。蚂蚁在这个图中行走以构造问题的解。文献中的蚁群算法的主要区别在于蚂蚁释放信息素的方式和启发信息的定义。假定 s 是一个解，且 $S=\{o_1,o_2,\cdots,o_{|S|}\}$ 是被选物品集，其中 $|S|$ 是 S 的基数。它们采用的信息素释放方式如下。

(1) 第一种方法是把信息素释放在被选物品上。在这种方式中，信息素用来表征对物品的偏好。它的基本思想是增加被选物品的偏好使得这些物品在以后的解构造过程中更可能被选中。

(2) 第二种方法是把信息素释放到连接每对先后被选物品(o_u,o_{u+1})的边上。其基本思想是当上一个被选物品是 o_u 时，增大选择 o_{u+1} 的可能性。在这种方式中，信息素表示从一个顶点(物品)出发到下一个特定顶点(物

品)的偏好。

(3) 第三种方法是把信息素释放到连接每对被选取物品的边上。其基本思想是增加同时选取 S 中任意两个物品的可能性。

至于启发信息,文献[9]和文献[11]中提出了一种动态启发信息,即在算法运行过程中启发信息依赖于当前已构造的部分解,其数学表达式为

$$\eta_{S_k}(j) = \frac{p_j}{\sum_{i=1}^{m} r_{ij} / d_{S_k}(i)} \tag{4-4}$$

其中 $d_{S_k}(i) = b_i - \sum_{g \in S_k} r_{ig}$,$S_k$ 是已被选中的物品组成的集合。

而文献[10]中提出了3种静态启发信息,即启发信息在算法运行过程中保持不变。它们分别如下:

$$\text{第一种启发信息:} \eta_j = p_j^{d_1} \tag{4-5}$$

其中 d_1 是参数。

$$\text{第二种启发信息:} \eta_j = \begin{cases} p_j^{d_1} / s_j^{d_2}, & s_j \neq 0 \\ p_j^{d_1}, & s_j = 0 \end{cases} \tag{4-6}$$

其中 $s_j = \max_{1 \leqslant i \leqslant m}(r_{ij})$,$d_1$ 和 d_2 是参数。

$$\text{第三种启发信息:} \eta_j = \begin{cases} p_j^{d_1} / s_j^{d_2}, & s_j \neq 0 \\ p_j^{d_1}, & s_j = 0 \end{cases} \tag{4-7}$$

其中 $s_j = \sum_{i=1}^{m} r_{ij}$,$d_1$ 和 d_2 是参数。

4.3 算法描述

4.3.1 算法的基本思想

早在1996年,Dorigo就指出蚂蚁系统可能出现早熟收敛[12],即算法会过早地收敛到一个局部最优解,而不能找到新的解。此后如何避免早熟收敛一直是蚁群算法研究的一个热点。最大最小蚂蚁系统(MMAS)提供了一种简便却很实用的方法,它通过设置信息素的上下界来避免早熟收敛。显然,信息素的上下界对于算法性能起着至关重要的作用。如果信息素的上下界差异过小,蚂蚁的搜索行为可能过度多样化;而如果信息素的上下界差异太大,又可能出现停滞。因此,信息素上下界的选取需要考虑搜索的多样性(diversification)和强化性(intensification)。

在MMAS中,信息素的上界一般设置为信息素的最大渐近估计值,同

题的难点在于信息素下界的设置。Stützle 和 Hoos 建议设置一个固定的信息素下界，其基本思想是：在信息素都收敛到信息素的上界或下界时，构造最优解的概率小于某个阈值。不过这种方法假设启发信息的影响可以忽略。对于很多问题来说，启发信息对蚁群算法的性能有着重要影响。虽然平滑机制能减弱算法对信息素下界的敏感度，但是 Stützle 和 Hoos 没有提供判断算法收敛的一般方法，这为合理地使用平滑机制带来不便。注意到在 MMAS 的更新规则中，只有构造了最优解的蚂蚁进行信息素更新。如果最优解对应的信息素与其他解元素对应的信息素差异过大，就可能造成在以后的构造过程中，最优解对应的解元素被选中的概率过大，从而出现构造的解与最优解的差异过小。如果能避免这种情形，就能避免早熟收敛。本章提出的方法通过修正信息素的下界来避免停滞，从而起到平衡搜索多样性和强化性的目的。由于这一方法具有自适应性，我们称之为自适应性方法，相应地，所提出的算法被称为自适应最大最小蚂蚁系统(AMMAS)。

AMMAS 与 MMAS 的主要区别在于信息素下界的选取。在算法中，只有最优蚂蚁才能更新信息素(记 s^{best} 为最优解)，它可能是当前最优解 s^{gb}，也可能是本次迭代的最优解 s^{ib}。在信息素更新后，最优解对应的信息素增加，在下次迭代中，最优解对应的解元素被选取的概率可能增大。如果能避免蚁群构造的解与 s^{best} 的差异过小，就可以保持搜索的多样性。因此，研究的关键点是衡量这个差异。

首先给出一个量来衡量两个解之间的差异。在蚁群算法中，蚂蚁依据信息素释放方式，在其构造的解 s 对应的边或者点上释放信息素。以后，称这些边或者点组成的集合为解 s 的**释放集**。由释放集可以给出一个衡量两个解之间差异的量。

定义 1(差异量) 给定两个解 s_a 和 s_b 并且其对应的释放集分别为 S_a 和 S_b，称 $D(s_a,s_b)=|S_a\oplus S_b|$ 为 s_a 和 s_b 的**差异量**，其中 $S_a\oplus S_b=S_a\cup S_b-S_a\cap S_b$，它是集合 S_a 和 S_b 的对称差，$|S_a\oplus S_b|$ 是 $S_a\oplus S_b$ 的基数。

差异量具有以下性质：① $D(s_a,s_b)\geqslant 0$；② $D(s_a,s_b)=D(s_b,s_a)$；③ $D(s_a,s_b)\leqslant D(s_a,s_c)+D(s_b,s_c)$，这是因为 $S_a\oplus S_b\subseteq(S_a\oplus S_c)\cup(S_b\oplus S_c)$。

依据差异量，可以定义平均差异量衡量蚁群构造的解与 s^{best} 的差异程度。

定义 2(平均差异量) 设 s^i $(1\leqslant i\leqslant n_a)$ 是蚁群构造的解，称 $\text{avg}_D=\sum_{i=1}^{n_a}D(s^i,s^{\text{best}})/n_a$ 为**平均差异量**，其中 n_a 为蚂蚁数。

需要指出的是，s^{best} 为上一次迭代中用来更新信息素的最优解，而

$s^i(1 \leqslant i \leqslant n_a)$是蚁群在本次迭代中构造的解。由于蚂蚁构造的解 s 是一个随机变量,下面考虑 $D(s, s^{\text{best}})$ 的期望 $E(D(s, s^{\text{best}}))$。由概率统计知识可知:

$$\text{avg}_D \to E(D(s, s^{\text{best}})), \quad 当\ n_a \to +\infty\ 时 \tag{4-8}$$

也就是说,avg_D 是 $E(D(s, s^{\text{best}}))$ 的一个估计,显然它是一个无偏估计量[13],以后称 $E(D(s, s^{\text{best}}))$ 为**期望差异量**。

如果平均差异量很小,这表明蚁群的搜索行为过度强化。在这种情形下,增大信息素下界可以使蚁群的搜索多样化。具体而言,在 AMMAS 中,信息素下界按照如下方式选取:

当找到一个新的当前最优解 s^{gb} 时,信息素的下界 $\tau_{\min}$ 被重新设置为一个很小的值。此后,如果平均差异量过小,可根据如下方式修正信息素的下界:

$$如果\ \text{avg}_D < \gamma, 则\ \tau_{\min} := \lambda \tau_{\min}。$$

其中,γ 是一个正参数$(1 < \gamma < |S^{gb}|)$,S^{gb} 是 s^{gb} 对应的释放集,$\lambda(\lambda > 1)$是一个正参数。

这种方法的一个特点是避免完全丢弃蚁群保存在信息素轨迹中的信息。在后续各节中,将分析这种方法对于算法性能的影响。

AMMAS 的基本流程是:在每一代,每只蚂蚁构造一个解,然后按照信息素更新规则来更新信息素。下面首先定义信息素和启发信息,再给出解的构造过程和信息素更新规则,最后给出了局部搜索。

4.3.2 信息素和启发信息的定义

多维背包问题是一个典型的子集问题。在子集问题中优化目标是从原物品集中选取一组最优可行物品。其他子集问题有约束满足问题(constraint satisfaction problem)[14]、装箱和下料问题(bin packing and cutting stock)[15]、加权边 k-势树问题(the edge-weighted k-cardinality tree problem)[16]以及最大团问题(the maximum clique problem)[17]等。与旅行商问题相比,子集问题关注于选取哪些物品而不注重物品被选取的次序[18]。考虑到计算时间和计算结果,AMMAS 采用 4.4 节中的第一种信息素释放方式,即蚂蚁将信息素释放到每个顶点(物品)上。在这种方式中,每个物品 j 具有信息素 $\tau(j)$。

除信息素之外,启发信息是影响解构造的另一个重要因素。算法中的启发信息是式(4-9)定义的伪效用率(pseudo-utility ratio)[3]

$$\forall j \in J, \quad \eta(j) = \frac{p_j}{\sum_{i=1}^{m} w_i r_{ij}} \tag{4-9}$$

其中,w_i 是第 i 个约束在原问题松弛线性规划的影子价格(对偶变量),分母表示物品 j 的紧致性(tightness)。由式(4-9)可见,蚂蚁偏好于利润高且紧致性小的物品。

4.3.3 解的构造

在构造解时,蚂蚁用一个 n 维向量标记物品是否被选取,它对应于一个解。这个向量的每个分量都初始化为0。在第 k 构造步,蚂蚁依据式(4-10)定义的概率选择物品

$$P(c_k = j \mid \tau) = \begin{cases} \dfrac{\tau(j)^{\alpha}\eta(j)^{\beta}}{\sum_{u \in U_k} \tau(u)^{\alpha}\eta(u)^{\beta}}, & j \in U_k \\ 0, & \text{其他} \end{cases} \tag{4-10}$$

其中,$U_k(1 \leqslant k \leqslant n)$ 是由满足约束且未被选择的物品组成的集合。$\alpha,\beta(\alpha > 0,\beta > 0)$ 是参数,它们控制信息素和启发信息的相对重要性。由式(4-10)可知,蚂蚁偏好于那些具有较高信息素和启发信息的物品。假设物品 j 被选取,则向量的第 j 个分量设置成1。构造过程在 U_k 为空集时终止。

在解构造过程中,可选物品集的确定对算法速度有重要影响。因为 $r_{ij} \geqslant 0$,所以一旦某个物品违反了约束,它在以后的构造过程都是不可选的,这就是说 $U_k \supset U_{k+1}$。假设在第 k 构造步,如果物品 $o \in U_k$ 满足 $r_{io} \leqslant b_i - \sum_{j=1}^{n} r_{ij} x_j (i = 1, \cdots, m)$,则该物品就是可选物品。注意到保存 $b_i - \sum_{j=1}^{n} r_{ij} x_j$ 可提高计算速度。

4.3.4 信息素的更新规则

在每个蚂蚁构造完一个解后,只使用最优解更新信息素。具体而言,信息素按照以下方式更新

$$\forall u \in J, \quad \tau(u)^{l+1} = \rho\tau(u)^{l} + \Delta\tau(u) \tag{4-11}$$

$$\text{如果 } \tau(u)^{l+1} < \tau_{\min}, \quad \text{则 } \tau(u)^{l+1} = \tau_{\min} \tag{4-12}$$

$$\text{如果 } \tau(u)^{l+1} > \tau_{\max}, \quad \text{则 } \tau(u)^{l+1} = \tau_{\max} \tag{4-13}$$

其中,$\tau(u)^l$ 是物品 u 在第 l 代时的信息素,ρ 是信息素保持率(pheromone persistence)($1-\rho$ 是信息素的挥发率)。令 $\tilde{s}^{\text{best}}$ 为最优解,它可能是当前最优解 s^{gb},也可能是本次迭代的最优解 s^{ib}。如果 $\tilde{s}^{\text{best}}$ 的第 u 个分量等于1,则

$\Delta\tau(u)$等于$g(s^{\mathrm{best}})$，其中$g(x)=1\Big/\sum_{j=1}^{n}p_j(1-x_j)$；否则$\Delta\tau(u)$为0。$\tau_{\max}$和$\tau_{\min}$分别是信息素的上下界。在信息素更新后，那些被最优蚂蚁选中的物品将接受更多的信息素，因此它们的偏好度(desirability)增加了。信息素的上界被设置为$g(s^{gb})/(1-\rho)$。在4.6节将详细讨论信息素下界的选取。

4.3.5　局部搜索

局部搜索能有效避免算法陷于局部极值点，因此它常用来改进蚁群算法。本章采用的局部搜索的基本思想为：用任意两个未被选物品替换一个被选物品，每次(可行的)替换获得一定的收益，直到找到收益最大的替换。为了提高局部搜索的计算速度，可以对所有物品按照收益进行降序排列。由于在确定第二个未被选物品时，如果本次替换的收益小于当前最大收益，就不用尝试其他未被选物品。因此降序排列能起到减少计算量的目的。

为区别起见，称具有局部搜索的算法为混合AMMAS(AMMAS+ls)，它按照以下方式运行：在每一代，一旦某个蚂蚁构造了一个解，用局部搜索来改进这个解。

4.4　信息素下界的选取

4.4.1　Stützle 和 Hoos 法的分析

在文献[19]中，Stützle 和 Hoos 建议将信息素的下界选取为

$$\tau_{\min}=\varepsilon\tau_{\max} \tag{4-14}$$

其中$\varepsilon=(1-\sqrt[n]{P_{\mathrm{best}}})/((avg-1)\sqrt[n]{P_{\mathrm{best}}})$，$avg$等于$n/2$，$P_{\mathrm{best}}(0<P_{\mathrm{best}}<1)$是参数。简单起见，称这种方法为SH法(SH method)。

在应用该方法时，关键在于选取合适的P_{best}。为了分析P_{best}对于蚂蚁搜索能力的影响，采用以下两个指标[18]。

(1) 相似率(similarity ratio)：这个指标被广泛用于量化多样性，并且它已经在进化算法中应用[20]。本文考虑文献[18]中提出的指标

$$\frac{\sum_{j=1}^{n}\left(\sum_{i=1}^{n_a}s_j^i\cdot\left(\sum_{i=1}^{n_a}s_j^i-1\right)\right)}{(n_a-1)\cdot\sum_{j=1}^{n}\sum_{i=1}^{n_a}s_j^i} \tag{4-15}$$

由式(4-15)可见，当所有构造的解相同时，相似率为1；而如果蚂蚁构造的解完全不同时，相似率为0。

(2) 重抽样率(re-sampling ratio):它用来刻画算法在搜索空间中抽样的有效性。令 DiffNum 为到目前为止蚂蚁构造的不同解的数目,TotalNum 为到目前为止蚂蚁构造的解的总数。重抽样率定义为

$$\frac{\text{TotalNum} - \text{DiffNum}}{\text{TotalNum}} \tag{4-16}$$

重抽样率接近于 0 意味着蚁群构造的解差异大,搜索能力强;而重抽样率接近于 1 意味着蚁群构造的解差异小,搜索接近停滞。

图 4-1 给出了相似率变化曲线。即使 P_{best}取为 0.005 这样一个很小的量,相似率增加得很快,在 150 代达到 0.98。图 4-2 给出了重抽样率变化曲线。在 P_{best}取为 0.005 时,重抽样率增加得也很快,在 150 代达到 0.36。这表明蚂蚁在一个很小的区域搜索并且 36% 的解已在以前的迭代中被蚂蚁构造过。在这种情形下,必须使蚂蚁的搜索行为多样化。我们知道启发信息在构造解时起着很重要的作用,而式(4-14)没有考虑启发信息。这就给如何合适地选取 P_{best}带来困难。特别地,当这种方法用到不同规模的问题时,选取这样一个重要的参数将是很不方便的。

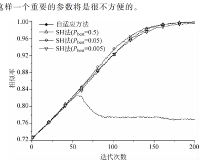


图 4-1 相似率的演变(自适应方法和 SH 法在算例 10.100.00 上的平均结果)
参数设置为:$n_a=50$,$\alpha=1$,$\beta=20$,$\rho=0.95$,$\gamma=8$,$\lambda=2$,$P_{best}=0.9$

4.4.2 自适应方法

由信息素的释放方式可见,任意两个解之间的差异量即为解之间的汉明(Hamming)距离(记为 d)。设上一次迭代中用来更新信息素的最优解为 s^{best},则平均差异量为





4.5 实验分析

为了检验 AMMAS 解码策略的性能，采用文献[3]中提供的算例进行实验。每个问题表示为 m.n.i，其中 m 和 n 分别是约束和物品的数目，i 表示该算例在具有相同数目的物品和约束的算例组中的序号。根据 Stützle 和 Hoos 的建议，问题分量概率调整为或是近似解的本次迭代最优解来更新信息素，令 f^{gb} 表示每隔 f^{gb} 代，f^{gb} 被用来更新信息素。调整策略为：在前 10 代，f^{gb} 为 3，从第 11 代到第 25 代，f^{gb} 为 2，从第 25 代起，f^{gb} 为 1。信息素的初始值为最大信息素值。为了统计分析，对每个算例测试 50 次。

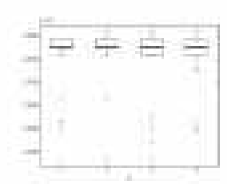
4.5.1 解的评价

对于随机算法，评价计算结果都是一个难点。虽然算法的收敛性保证了算法在无穷大时间内总能找到最优解，但是实际计算时间是有限的。本章采用文献[3]中的方法，它先通过求解多维背包问题的线性规划松弛问题获得该问题的上界，再通过比较随机算法求解的结果与问题的上界的差距(gap)进行评价。由于文献[3]中的 GA 算法计算结果与问题上界的差距很小，因此 GA 是一个好的算法。所以与 GA 算法进行比较，能判断 AMMAS 的计算结果好坏。

4.5.2 参数选取

和文献[3]中的终止条件一样，AMMAS 在检查 10000 个解之后停止。下面用数值实验分析参数的影响。在实验中，参数的默认值为：$n_a=10$，$\alpha=1$，$\beta=10$，$\rho=0.95$，$\gamma=8$，$\lambda=2$，$P_{best}=0.5$。在进行实验时，只一个参数变化，其他参数保持不变，被测试的参数取值为 $n_a\in\{10, 20, 50, 100\}$，$\alpha\in\{0.5, 1, 2, 3\}$，$\beta\in\{1, 5, 10, 20, 40\}$，$\rho\in\{0.7, 0.8, 0.9, 0.95, 0.98\}$，$\gamma\in\{2, 4, 8, 16\}$，$\lambda\in\{1.5, 2, 2.5, 3\}$，$P_{best}\in\{0.5, 0.7, 0.8, 0.9\}$。我们只选择 30.100.00 为例进行评估。对于每个参数值，有 50 个样本(即 50 个最优值)。数据用箱形图(box plot)表示(本文中的箱形图使用 Matlab 生成)。在箱形图中，中间的横线表示中位数，上下横线分别为样本的上下四分位数，箱形宽度和数据的点数成比例，箱形的上部向下延伸的虚线称为"触须"，直到最外侧数值，样本的最大值为上触须的顶端，样本的最小值为下触须的底端，距离箱形顶部或底部超过 1.5 倍四分位距的值称为异常值，在图中用空心点表示。每个图的纵坐标为对应运行得到的总收益。

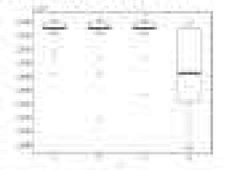


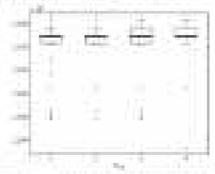












个算例为一组，每组用记号表示为 $m.n$。对于 5.500、10.500 和 30.500，最大运行时间分别是 100 秒、200 秒、400 秒(CPU 为 2.8GHz)。前期实验表明，算法能在相应的时间内获得的解是令人满意的。需要说明的是，为了公平比较混合 AMMAS 和 AMMAS，将最大运行时间作为停止条件。

首先，分析局部搜索对算法的影响。表 4-3 给出了 AMMAS 和 AMMAS+ls 的实验结果，并且给出了不考虑信息素时(此时 $\alpha=0$)AMMAS+ls 的实验结果。由结果可见，局部搜索能有效改善 AMMAS 的性能，AMMAS+ls 在所有算例中都取得最好的解；并且，使用信息素能引导蚂蚁找到更好的解，AMMAS 的性能优于没有使用信息素的混合算法。

表 4-3 AMMAS+ls 和 AMMAS 的实验结果

算例	AMMAS+ls($\alpha=1$)		AMMAS+ls($\alpha=0$)		AMMAS	
	最优值	平均值	最优值	平均值	最优值	平均值
5.500.00	**120148**	**120111**	119860	119648	120116	120056
5.500.01	**117879**	**117841**	117494	117360	117857	117786
5.500.02	**121131**	**121097**	120708	120526	121109	121043
5.500.03	**120804**	**120776**	120473	120293	120785	120715
5.500.04	**122319**	**122303**	121988	121812	122319	122254
5.500.05	**122024**	**121991**	121695	121562	121992	121936
5.500.06	**119127**	**119093**	118735	118614	119096	119043
5.500.07	**120568**	**120525**	120209	120121	120536	120472
5.500.08	**121575**	**121537**	121095	120961	121551	121479
5.500.09	**120717**	**120678**	120334	120192	120692	120627
5.500.10	**218428**	**218397**	218111	217943	218400	218344
5.500.11	**221202**	**221168**	220808	220668	221191	221117
5.500.12	**217534**	**217513**	217150	217039	217528	217459
5.500.13	**223560**	**223547**	223236	223136	223560	223499
5.500.14	**218966**	**218956**	218675	218528	218962	218905
5.500.15	**220530**	**220497**	220228	220132	220496	220455
5.500.16	**219989**	**219974**	219632	219519	219987	219924
5.500.17	**218194**	**218171**	217848	217758	218180	218124
5.500.18	**216963**	**216948**	216634	216551	216958	216904
5.500.19	**219719**	**219694**	219367	219188	219704	219657
5.500.20	**295828**	**295809**	295628	295485	295828	295764
5.500.21	**308086**	**308069**	307893	307805	308077	308023
5.500.22	**299796**	**299781**	299620	299527	299796	299738
5.500.23	**306480**	**306467**	306338	306238	306480	306427
5.500.24	**300342**	**300334**	300175	300076	300334	300280

续表

算　例	AMMAS＋ls(α=1)		AMMAS＋ls(α=0)		AMMAS	
	最优值	平均值	最优值	平均值	最优值	平均值
5.500.25	**302571**	**302556**	302421	302327	302560	302525
5.500.26	**301329**	**301317**	301157	301082	301325	301278
5.500.27	**306454**	**306426**	306269	306200	306422	306388
5.500.28	**302828**	**302810**	302671	302566	302809	302765
5.500.29	**299906**	**299894**	299756	299656	299902	299845

注：物品数为500，约束数为5。平均值用四舍五入取整。

最后将AMMAS＋ls与两个优秀算法相比较。由于这两个算法只给出了每个算例的最大总利润，这里我们仅比较最大总利润。表4-4给出了3个算法的实验结果，表4-5、表4-6分别给出了AMMAS＋ls在30个10.500和30.500算例中获得的最大值。与GA相比，AMMAS＋ls在每一组都找到了更好的解；而与z^*相比，AMMAS＋ls在8组中找到了更好的解。

表4-4　AMMAS＋ls、GA和z^*实验结果比较

算　例　组	Tightness 率	AMMAS＋ls	GA	z^*
5.500	0.25	**120629**	120616	120623
5.500	0.5	**219509**	219503	219507
5.500	0.75	**302362**	302355	302360
10.500	0.25	**118603**	118566	118600
10.500	0.5	**217309**	217275	217298
10.500	0.75	**302588**	302556	302575
30.500	0.25	115541	115470	**115547**
30.500	0.5	**216223**	216187	216211
30.500	0.75	**302406**	302353	302404

注：计算结果用四舍五入取整。

表4-5　AMMAS＋ls求得的最大值

算　例	最大值	算例	最大值	算例	最大值
10.500.00	117784	10.500.10	217353	10.500.20	304353
10.500.01	119198	10.500.11	219041	10.500.21	302371
10.500.02	119196	10.500.12	217797	10.500.22	302416
10.500.03	118813	10.500.13	216868	10.500.23	300757
10.500.04	116487	10.500.14	213816	10.500.24	304367
10.500.05	119454	10.500.15	215086	10.500.25	301796
10.500.06	119813	10.500.16	217931	10.500.26	304949

续表

算　例	最大值	算例	最大值	算例	最大值
10.500.07	118312	10.500.17	219984	10.500.27	296450
10.500.08	117779	10.500.18	214346	10.500.28	301331
10.500.09	119197	10.500.19	220865	10.500.29	307089

注：每个算例的物品数500，约束数为10。

表 4-6　AMMAS＋ls 求得的最大值

算例	最大值	算例	最大值	算例	最大值
30.500.00	115942	30.500.10	218034	30.500.20	301643
30.500.01	114732	30.500.11	214626	30.500.21	300014
30.500.02	116613	30.500.12	215903	30.500.22	305062
30.500.03	115263	30.500.13	217862	30.500.23	302001
30.500.04	116487	30.500.14	215622	30.500.24	304416
30.500.05	115734	30.500.15	215829	30.500.25	296962
30.500.06	114107	30.500.16	215883	30.500.26	303328
30.500.07	114252	30.500.17	216448	30.500.27	306944
30.500.08	115271	30.500.18	217333	30.500.28	303158
30.500.09	117011	30.500.19	214690	30.500.29	300531

注：算例的物品数500，约束数为30。

4.6　小结

本章首先用释放集的对称差作为衡量两个解之间差异的指标，并提出平均差异量来衡量蚁群构造解与上一代中用来更新信息素的最优解之间的差异。在此基础上，提出了自适应最大最小蚂蚁系统。它在平均差异量过小时自适应地修正信息素下界。针对多维背包问题的实验表明，这种方法能有效地平衡搜索的多样性和强化性，从而达到改善算法性能的目的。与其他蚁群算法以及现有优秀算法相比，本章所提出的算法能求得问题令人满意的解。

在应用自适应方法来选取信息素下界时，要确定信息素下界的初始值、参数 γ 和 λ。一般地，信息素下界的初始值通过将 P_{best} 设置为一个较大的值（通常取值为0.9）实现；参数 γ 在 $(1,|S^{gb}|)$ 之间选取一个较小的值；λ 在 $(1,3]$ 之间取值（通常取值为2）。

在蚁群算法中，解的构造相当于在解空间抽样。利用这些抽样信息，可以提出不同的统计量。自适应最大最小蚂蚁系统利用平均差异量对蚁群搜索行为进行评价，并且通过修正信息素下界改善算法性能。由于采用了自

适应方法，它为选取信息素下界这一重要参数提供了可行方法。

参考文献

[1] Shih W. A branch and bound method for the multiconstraint zero-one knapsack problem[J]. Journal of the Operational Research Society，1979，30：369～378.

[2] Gavish B，Pirkul H. Allocation of databases and processors in a distributed computing System[C]. Management of Distributed Data Processing，Akoka J. eds. North-Holland，1982，pp. 215～231.

[3] Chu P C，Beasley J E. A genetic algorithm for the multidimensional knapsack problem[J]. Journal of Heuristics，1998，4：63～86.

[4] Gavish B，Pirkul H. Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality [J]. Mathematical Programming，1985，31：78～105.

[5] Osorio M A，Glover F，Hammer P. Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions [R]. Technical report，Hearin Center for Enterprise Science. Report HCES-08-00，2000.

[6] Glover F，Kochenberger GA. Critical event tabu search for multidimensional knapsack problems[C]. Osman，I H，Kelly J P. ed. Metaheuristics：Theory and Applications. MA：Kluwer Academic Publishers，1996：407～427.

[7] Hanafi S，Freville A. An efficient tabu search approach for the 0-1 multidimensional knapsack problem [J]. European Journal of Operational Research，1998，106：659～675.

[8] Vasquez M，Hao J K. A hybrid approach for the 0-1 multidimensional knapsack problem[C]. Proceedings of the 13th International Joint Conference on Artificial Intelligence，2001,1：328～333.

[9] Alaya I，Solnon C，Ghdira K. Ant algorithm for the multidimensional knapsack problem[C]. International Conference on Bioinspired Optimization Methods and their Applications，2004：63～72.

[10] Fidanova S. ACO algorithm for MKP using various heuristic information[C]. Dimov，I. et al. ed. The 5th International Conference on NMA，vol. 2542，Lecture Notes in Computer Science，Berlin，Germany，2002：438～444.

[11] Leguizamon G，Michalewicz Z. A new version of ant system for subset problem [C]. Proceedings Congress on Evolutionary Computation，1999：1459～1464.

[12] Dorigo M，Maniezzo V，Colorni A. Ant system：Optimization by a colony of cooperating agents[J]. IEEE Transactions on System Man，and Cybernetics-Part B，1996，26：29～41.

[13] Devore J L. Probability and statistics：For engineering and the sciences[M]. CA：Duxbury Press，2000.

[14] Solnon C. Ants can solve constraint satisfaction problems[J]. IEEE Transactions on Evolutionary Computation，2002，6(4)：347～357.

[15] Levine J，Ducatelle F. Ant colony optimization and local search for bin packing and cutting stock problems[J]. Journal of the Operational Research Society，2004，55：705～716.

[16] Blum C，Blesa M J. New metaheuristic approaches for the edge-weighted k-cardinality tree problem[J]. Computers and Operations Research，2005，32：1355～1377.

[17] Solnon C，Fenet S. A study of ACO capabilities for solving the maximum clique problem[J]. Journal of Heuristics，2006，12：155～180.

[18] Solnon C，Bridge D. An ant colony optimization meta-heuristic for Subset selection problems[C]. System Engineering using Particle Swarm Optimization，Nedjah N，Mourelle L. Eds.，NY：Nova Science publisher，2006，pp. 7～29.

[19] Stützle T，Hoos H H. MAX-MIN ant system[J]. Future Generation Computer Systems，2000，16(8)：889～914.

[20] Morrison R W，De Jong K A. Measurement of population diversity[C]. Collet P. et al. ed. the 5th International Conference on EA，vol. 2310，Lecture Notes in Computer Science，2001，pp. 31～41.



定向问题

5.1 问题描述

定向问题来源于定向运动，在车辆路径与生产调度等生产生活中经常涉及这个问题[1]。为求解该问题，人们提出了许多精确算法。Hayes 和 Norman[2]采用动态规划法，Laporte 和 Martello[3]提出了线性规划松弛法和分支定界法的混合算法，Leifer 和 Rosenwein[4]提出线性规划松弛法和割平面法的混合算法，Fischetti 等[5]用分支定界法来求解。但是，定向问题是一个 NP-hard 问题，人们试图用启发式算法来求解该问题。Tsiligirides[6]提出了一类确定性启发算法和随机性启发算法，Ramesh 等[7]提出了四步法，文献[8]中的五步法是较有影响的一类启发式算法。另外，人工神经网络[9]也在定向问题中得到了较好的应用。最近 Liang 和 Smith[10]用蚁群算法来求解定向问题，其算法具有以下两个特点：①在 ACS 框架下构造问题的解并且更新信息素；②使用罚函数法来处理约束，并且采用了局部搜索改进蚂蚁构造的解。

定向问题的数学模型描述如下：给定图 $G=(V,E)$，点集 $V=\{1,2,\cdots,n\}$，边集 $E=\{(i,j)\mid i,j\in V\}$。每个点 i 具有一定收益 $r_i(r_i\geqslant 0)$，且 $r_1=r_n=0$(点1 称为起点，点 n 称为终点)。任意两点 i 和 j 的距离为 c_{ij}。每经过一个点可获得该点对应的收益且每个点最多只能经过一次。定向问题的

优化目标是寻找一条从点 1 出发到点 n 终止的路径，使得所获得的总收益最大且路径长度不超过 $T_{\max}$。定向问题的数学描述为[10]

$$\max \sum_{i=1}^{n}\sum_{j=1}^{n} r_i x_{ij} \tag{5-1}$$

$$\text{s.t.} \quad \sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1 \tag{5-2}$$

$$\sum_{i=2}^{n-1} x_{ik} = \sum_{j=2}^{n-1} x_{kj} \leqslant 1, \quad k = 2,3,\cdots,n-1 \tag{5-3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \leqslant T_{\max} \tag{5-4}$$

$$\sum_{i,j\in S} x_{ij} \leqslant |S|-1, \quad \forall S\subseteq V, |S| \geqslant 3 \tag{5-5}$$

$$x_{ij} \in \{0,1\}, \quad i,j = 1,2,\cdots,n \tag{5-6}$$

其中 $x_{ij}(1\leqslant i,j\leqslant n)$是二值变量，如果路径中包括了边$(i,j)$，$x_{ij}$ 等于 1，否则为 0。约束(5-2)确保每个路径(即解)都从点 1 出发到点 n 终止；约束(5-3)限定除点 1 和点 n 外其他各点最多只能经过一次；约束(5-4)确保路径的长度不超过 $T_{\max}$；约束(5-5)确保任意边最多只能经历一次；约束(5-6)限定所有变量只能取值为 0 或 1。

5.2 算法描述

实际上，如果定向问题中的 $T_{\max}$为无穷大且点 1 和点 n 相同，那么定向问题可以看成 TSP 问题[8]。本章提出的算法与文献[11]中求解 TSP 的蚁群算法相似，边(i,j)上的信息素为 $\tau(i,j)$，并且蚂蚁将信息素释放到所经过的路径上。其主要流程是：

步骤 1，设定参数，并且初始化信息素；

步骤 2，按照路径选择规则构造问题的解；

步骤 3，按照信息素更新规则更新信息素；

步骤 4，判断停止条件是否满足，若满足，算法终止，否则返回到步骤 2。

下面首先给出启发信息的定义，再讨论解的构造，最后讨论信息素的更新规则。

5.2.1 启发信息的定义

本文采用文献[6]提出的量来定义启发信息。边(i,j)的启发信息定义为

$$\eta(i,j) = (r_j / c_{ij})^4 \tag{5-7}$$

由式(5-7)可知,那些具有较高收益并且距离点 i 较近的点,偏好度较高。

5.2.2 解的构造

由约束(5-4)可知,违反式(5-8)的点一定是不可行点[8]。

$$c_{i1} + c_{in} \leqslant T_{\max}(2 \leqslant i \leqslant n-1) \tag{5-8}$$

事先去掉这些不可行点,能够减少所需要考虑的点数。不失一般性,假定所有点都满足式(5-8)。

为了说明构造过程,采用以下记号。

- T:当前(未完成)路径;
- O:路径 T 包含的点集;
- $L(T)$:路径 T 的长度;
- c_k:第 k 个构造步时选取的点。

在构造解时,每只蚂蚁从点 1 出发,在第 $k+1$ 构造步,它依式(5-9)定义的概率从可行点集 $C_{c_k} = \{j \in V \setminus (\{1,n\} \cup O) \mid L(T) + c_{c_k j} + c_{jn} \leqslant T_{\max}\}$ 中选取一个点

$$p_{c_k j} = \begin{cases} \dfrac{\tau(c_k,j)^{\alpha} \cdot \eta(c_k,j)^{\beta}}{\sum\limits_{u \in C_{c_k}} \tau(c_k,u)^{\alpha} \cdot \eta(c_k,u)^{\beta}}, & j \in C_{c_k} \\ 0, & \text{其他} \end{cases} \tag{5-9}$$

直到可行点集为空,最后选取点 n,这样就完成一个解的构造。

5.2.3 信息素的更新规则

在所有蚂蚁都构造完一个解后,信息素按照以下规则更新:

$$\tau(u,v)^{l+1} = \rho\tau(u,v)^l + \Delta\tau(u,v) \tag{5-10}$$

$$\text{如果 } \tau(u,v)^{l+1} < \tau_{\min}, \quad \text{则 } \tau(u,v)^{l+1} = \tau_{\min} \tag{5-11}$$

$$\text{如果 } \tau(u,v)^{l+1} > \tau_{\max}, \quad \text{则 } \tau(u,v)^{l+1} = \tau_{\max} \tag{5-12}$$

$$\Delta\tau(u,v) = \begin{cases} f(s^{\text{best}}), & (u,v) \in s^{\text{best}} \\ 0, & \text{其他} \end{cases} \tag{5-13}$$

$$f(x) = 1 \Big/ \left(\sum_{i=1}^{n} r_i - \sum_{i=1}^{n} \sum_{j=1}^{n} r_i x_{ij} \right) \tag{5-14}$$

其中 $\tau(u,v)^l$ 是在第 l 代边 (u,v) 上的信息素。s^{best} 是最优解,它可能是当前最优解 s^{gb} 也可能是本次迭代最优解 s^{ib}。注意到如果式(5-14)的分母等于

0，此时最优解必然被找到，算法可终止，因此式(5-14)是有意义的。τ_{max}和τ_{min}分别是信息素的上下界。信息素的上界设置为$f(s^{gb})/(1-\rho)$。信息素下界按照如下的方式选取：当找到一个新的s^{gb}时，τ_{min}被初始化为一个很小的值。这可以通过将τ_{min}设置为$(1-\sqrt[n]{P_{best}})/((avg-1)\sqrt[n]{P_{best}})\tau_{max}$且$P_{best}$设置为一个很大的数(如$P_{best}>0.5$)来实现。此后，如果平均差异量过小，信息素的下界按照如下方式修正：

$$\text{如果 } avg_D<\gamma,\quad \text{则 } \tau_{min}:=\lambda\tau_{min} \tag{5-15}$$

其中$\gamma(1<\gamma<n)$、$\lambda(\lambda>1)$是参数。

在后续章节中，将研究差异量的性质，并且讨论平均差异量的计算方法。

5.3　差异量的性质

在定向问题中，差异量是一个距离。

性质1　$D(\cdot,\cdot)$是距离。

证明：给定解s_a、s_b和s_c，它们对应的释放集分别为S_a、S_b和S_c。由D的定义可知：①$D(s_a,s_b)\geqslant 0$；②$D(s_a,s_b)=0$当且仅当$s_a=s_b$；③$D(s_a,s_b)=D(s_b,s_a)$；④注意到$S_a\oplus S_b\subseteq(S_a\oplus S_c)\cup(S_b\oplus S_c)$，因此$D(s_a,s_b)\leqslant D(s_a,s_c)+D(s_b,s_c)$。综上所述，结论成立。

注意到，证明中的②在所有问题中不一定都成立。下面来分析差异量与信息素更新的关系。显然地，蚂蚁构造的解是一个随机变量。设在第t次迭代中，信息素向量为$\boldsymbol{\tau}_t$，用来更新信息素的最优解为s_t^{best}，在第$t+1$次迭代中信息素向量为$\boldsymbol{\tau}_{t+1}$。假设s是按照5.2.2节的构造方法生成的一个随机解。在不考虑信息素上下界的情况下，蚂蚁在信息素更新后重新构造最优解s_t^{best}的概率增大，即以下结论成立。

性质2　$P(D(s,s_t^{best})=0\mid\boldsymbol{\tau}_t)<P(D(s,s_t^{best})=0\mid\boldsymbol{\tau}_{t+1})$。

证明：假设s_t^{best}第k个点为$u_k(1\leqslant k<|s_t^{best}|$，其中$|s_t^{best}|$为$s_t^{best}$所包含的点数)，$u_{k+1}$为第$k+1$点，$u_{k+1}\in C_{u_k}$为可行点集，注意它在信息素更新前后是不变的。而依据解的构造过程以及概率选择规则式(5-9)可知，在第t次迭代选取u_{k+1}的概率为

$$p_{u_k u_{k+1}}=\begin{cases}\dfrac{\boldsymbol{\tau}(u_k,u_{k+1})^{\alpha}\cdot\eta(u_k,u_{k+1})^{\beta}}{\sum\limits_{u\in C_{u_k}}\boldsymbol{\tau}(u_k,u)^{\alpha}\cdot\eta(u_k,u)^{\beta}}, & u_{k+1}\in C_{u_k}\\ 0, & \text{其他}\end{cases} \tag{5-16}$$

而在第 $t+1$ 次迭代中，选取 u_{k+1} 的概率为

$$p'_{u_k u_{k+1}} = \begin{cases} \dfrac{\left(\boldsymbol{\tau}(u_k, u_{k+1}) + \dfrac{\Delta\tau(u_k, u_{k+1})}{\rho}\right)^{\alpha} \cdot \eta(u_k, u_{k+1})^{\beta}}{\sum\limits_{u \in C_{u_k}} \left(\boldsymbol{\tau}(u_k, u) + \dfrac{\Delta\tau(u_k, u)}{\rho}\right)^{\alpha} \cdot \eta(u_k, u)^{\beta}}, & u_{k+1} \in C_{u_k} \\ 0, & \text{其他} \end{cases} \tag{5-17}$$

依据不等式

$$\frac{a}{a+b} < \frac{a+\Delta}{a+b+\Delta} \quad a, b, \Delta > 0 \tag{5-18}$$

可知 $p'_{u_k u_{k+1}} > p_{u_k u_{k+1}}$，这证明了在信息素更新后，选择 s_t^{best} 的概率增加了。从而证明结论。

推论　如果在第 $t+1$ 次迭代中 s_t^{best} 对应边上的信息素为无穷大，则 $E(D(s, s_t^{\text{best}}) \mid \boldsymbol{\tau}_{t+1}) = 0$。

证明：类似于性质 2 的证明可知，$P(D(s, s_t^{\text{best}}) = 0 \mid \boldsymbol{\tau}_{t+1}) = 1$。依据期望的定义可知结论成立。

实际上，如果 s_t^{best} 对应边上信息素与其他边上的信息素的差异过大，此时期望差异量将会很小（接近于 0），蚂蚁的探索能力非常弱。而平均差异量是期望差异量的无偏估计，在平均差异量小于某个阈值时，可以通过修改信息素的下界来保持搜索的多样性。

5.4　平均差异量的计算

为计算平均差异量，关键在于计算任意两个解的差异量。设两个路径（解）$\boldsymbol{s}_a = (a_1, \cdots, a_u)$，$\boldsymbol{s}_b = (b_1, \cdots, b_v)$，其中 $a_1 = b_1 = 1$，$a_u = b_v = n$，u 和 v 分别是两个路径所经过的点数，不失一般性，假设 $u \geqslant v$。它们的释放集分别为

$$S_a = \{(a_1, a_2), \cdots, (a_{u-1}, a_u)\} \tag{5-19}$$

$$S_b = \{(b_1, b_2), \cdots, (b_{v-1}, b_v)\} \tag{5-20}$$

由差异量的定义可知 $D(\boldsymbol{s}_a, \boldsymbol{s}_b) = u + v - 2|S_a \cap S_b| - 2$，因此只需要计算 $|S_a \cap S_b|$。注意到 S_a 中除了 (a_1, a_2) 和 (a_{u-1}, a_u) 之外的元素都是对称的，即 $(a_i, a_{i+1}) = (a_{i+1}, a_i)$ $(1 < i < u)$。直接的方法是把 S_b 中的每个元素与 S_a 来比较，但这样计算代价较大，其计算复杂度为 $O(uv)$。实际上，可以先将 s_a 用两个 n 维数组 $\boldsymbol{P}$、$\boldsymbol{Q}$ 表示，其中 $\boldsymbol{P}[a_i] = a_{i+1}$，$\boldsymbol{Q}[a_{i+1}] = a_i$ $(1 \leqslant i < u)$，$\boldsymbol{P}$ 和 $\boldsymbol{Q}$ 其他的元素为 0。这个过程的计算复杂度为 $O(u)$。现在要判断 S_b 中的某个元素如 (b_2, b_3) 是否属于 S_a，只需要比较 $\boldsymbol{P}[b_2]$ 和 b_3 以及 $\boldsymbol{Q}[b_2]$

和 b_3 即可。计算两个解差异量的伪代码在图 5-1 中给出。

```
将解 s_a 用数组 P,Q 表示
u←s_a 释放集的基数;
v←s_b 释放集的基数;
d←0;
D(s_a,s_b)←0
for j = 1 to v−1 do
  if P[b_j]=b_{j+1}
     d←d+1;
  elseif Q[b_j]=b_{j+1}
     d←d+1;
  end if
end for
D(s_a,s_b)←u+v−2d−2
输出 D(s_a,s_b)
```

图 5-1　计算两个解的差异量的伪代码

由伪代码可见，$D(s_a,s_b)$的计算复杂度为 $O(v)+O(u)$。显然，$O(u)$和 $O(v)$都不大于 $O(n)$，因此，在最差情形下，差异量的计算复杂度不超过 $O(n)$，平均差异量的计算复杂度不超过 $O(n_a n)$。

例：考虑总点数为 7 的定向问题，一个解 $s_a=(1,2,3,7)$，另一个解 $s_b=(1,3,2,4,7)$，则 s_a 对应的向量 $\boldsymbol{P}=(2,3,7,0,0,0,0)$，$\boldsymbol{Q}=(0,1,2,0,0,0,3)$。依据差异量的定义可见，$D(s_a,s_b)=6$ 且 $S_a \cap S_b=\{(2,3)\}$。

5.5　实验分析

本文所提出的算法用 C++ 实现并且在 P4 3GHz PC 上测试。采用文献[8]中的算例集来测试算法性能。参数选取为：蚂蚁数 $n_a=20$，$\alpha=1$，$\beta=1$，$\rho=0.95$，$\lambda=2$，$\gamma=6$，$P_{\text{best}}=0.9$。每隔 10 代用 s^{gb} 来更新信息素，而在其他代中使用 s^{ib}。为使 AMMAS 与文献[10]中的算法构造解的个数一样，算法最大运行代数为 300。每个算例(记为 $n\times T_{\max}$)分别测试 20 次。在实验中，按照文献[8]的建议将路径的长度四舍五入到小数点第一位。

为了分析 AMMAS 的性能，首先将其与 MMAS 以及 MMAS＋ri 作比较，其中 MMAS＋ri 是在 MMAS 中加入信息素重新初始化机制，其具体实现是在平均差异量小于 γ 时，将信息素的值设置为信息素上界。实际上，在

平均差异量较小时，也可以重新初始化信息素，从而避免信息素的差异过大。在 MMAS 和 MMAS＋ri 中，按照文献[11]的建议将 P_{best} 设置为 0.05，其他参数与 AMMAS 的一样。

图 5-2 显示了 3 种算法应用到算例 64×80(这里 n 为 64，T_{max} 为 80)的总收益变化曲线(20 次平均)。由图 5-2 可见，在前 50 代，3 种算法的计算结果差异很小。其原因为：在 AMMAS 中，信息素的下界并没有被修正；而在 MMAS＋ri 中，信息素重新初始化机制也没有发挥作用，它们与 MMAS 工作机制类似。而在 50 代以后，特别是在 150 代以后，AMMAS 的计算结果最好，而 MMAS＋ri 优于 MMAS。这表明自适应机制和信息素重新初始化机制能起到改善算法性能的作用。

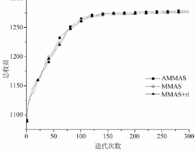


图 5-2　AMMAS、MMAS 和 MMAS＋ri 的总收益变化曲线
(在算例 64×80 上的结果)

图 5-3 显示了 3 种算法应用到算例 64×80(这里 n 为 64，T_{max} 为 80)的平均差异量的变化曲线。由图 5-3 可见，在 50 代之前，3 种算法的平均差异量差异较小。此时，3 种算法都有很好的探索能力。在 180 代后，MMAS 平均差异量很小(在 3.8 到 7.2 之间波动)，这意味着蚁群选择的边大多属于最优解，它们仅能探索到少量其他的边，而 AMMAS 的平均差异量较大。MMAS＋ri 通过重新初始化信息素也能避免平均差异量过小，在每次初始化后，平均差异量出现很大的波动。MMAS＋ri 与 AMMAS 的区别在于它完全丢弃了保存在信息素轨迹中的信息，这可能对算法性能带来不利影响。

表 5-1 和表 5-2 分别给出了两个最大的算例集(即钻石形(Diamond-

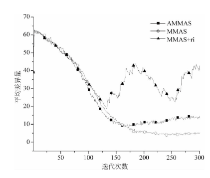


图 5-3 AMMAS、MMAS 和 MMAS+ri 的平均差异量变化曲线
(在算例 64×80 上的结果)

shaped)和方阵形(Square-shaped)算例集)中的实验结果(前一个算例集中的点数 n 为 64,共 14 个算例;后一个算例集中的点数 n 为 66,共 26 个算例)。表 5-1 和表 5-2 中给出了每个算例的最大总收益(最大值)和平均总收益(平均值),在表中第二列给出的是文献[8]中的算法(CGW)的实验结果,每一列最好的结果用粗体字给出。

表 5-1 AMMAS、MMAS 和 MMAS+ri 在 Diamond-shaped 数据集中的实验结果

算 例	CGW	AMMAS		MMAS		MMAS+ri	
	最大值	最大值	平均值	最大值	平均值	最大值	平均值
64×15	96	96	96.0	96	96.0	96	96.0
64×20	294	294	294.0	294	294.0	294	294.0
64×25	390	390	390.0	390	389.1	390	390.0
64×30	474	**486**	484.8	**486**	484.5	**486**	**485.4**
64×35	570	**576**	**569.1**	**576**	567.9	**576**	567.6
64×40	714	714	**710.7**	714	708.9	714	708.8
64×45	816	816	**806.1**	816	803.4	816	805.5
64×50	900	900	**895.8**	900	891.0	900	894.0
64×55	984	984	975.6	984	976.2	984	**977.1**
64×60	1044	**1062**	**1058.1**	**1062**	1055.4	**1062**	1056.3
64×65	1116	1116	**1114.8**	1116	1113.0	1116	1113.0

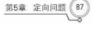


续表

算　例	CGW	AMMAS		MMAS		MMAS＋ri	
	最大值	最大值	平均值	最大值	平均值	最大值	平均值
64×70	1176	**1188**	1186.8	**1188**	1186.8	**1188**	**1187.1**
64×75	1224	**1236**	**1231.5**	**1236**	1230.3	**1236**	1229.1
64×80	1272	**1284**	**1278.0**	**1284**	1273.8	**1284**	1274.4

表 5-2　AMMAS、MMAS 和 MMAS＋ri 在 Square-shaped 数据集中的实验结果

算　例	CGW	AMMAS		MMAS		MMAS＋ri	
	最大值	最大值	平均值	最大值	平均值	最大值	平均值
66×5	10	10	10.0	10	10.0	10	10.0
66×10	40	**45**	45.0	**45**	45.0	**45**	45.0
66×15	120	120	120.0	120	120.0	120	120.0
66×20	195	**205**	**203.8**	**205**	202.3	**205**	**203.8**
66×25	290	290	290.0	290	290.0	290	290.0
66×30	400	400	400.0	400	400.0	400	400.0
66×35	460	**465**	462.0	**465**	461.8	**465**	**462.3**
66×40	575	575	575.0	575	575.0	575	575.0
66×45	650	650	**647.5**	650	646.5	650	647.0
66×50	730	730	730.0	730	729.8	730	730.0
66×55	825	825	**823.5**	825	821.5	825	823.3
66×60	915	915	914.8	915	914.8	915	914.8
66×65	980	980	980.0	980	980.0	980	980.0
66×70	1070	**1075**	1075.0	**1075**	1075.0	**1075**	1075.0
66×75	1140	1140	1140.0	1140	1140.0	1140	1140.0
66×80	1215	1215	1214.3	1215	1214.3	1215	**1214.8**
66×85	1270	1270	**1267.3**	1270	1266.3	1270	1266.8
66×90	1340	1340	**1340.0**	1340	**1340.0**	1340	1339.5
66×95	1380	**1395**	**1393.3**	**1395**	1391.3	**1395**	1392.5
66×100	1435	**1465**	**1464.5**	**1465**	1463.5	**1465**	1464.3
66×105	1510	**1520**	**1519.8**	**1520**	1519.0	**1520**	1519.0
66×110	1550	**1560**	**1560.0**	**1560**	1559.8	**1560**	**1560.0**
66×115	1595	1595	**1594.0**	1595	1592.5	1595	1593.3
66×120	1635	1635	**1635.0**	1635	**1635.0**	1635	1634.8
66×125	1655	**1670**	**1669.8**	**1670**	1669.0	**1670**	1667.3
66×130	1680	1680	**1679.8**	1680	**1679.8**	1680	1679.5



由表 5-1 和表 5-2 中的结果可知：3 种蚁群算法都能得到 15 个算例新的最大总收益。在平均总收益方面，AMMAS 的结果最好，MMAS＋ri 次之，这表明所提出的方法能有效地改善算法的搜索能力。并且，如果避免完全丢弃蚁群在以前迭代中积累的信息，可以改善算法的性能。在计算时间方面，本章提到的算法都能在 0.59 秒内求解每一个算例，这表明算法能较快地求解问题。另外，实验结果表明，AMMAS 计算平均差异量所花费的时间远小于算法中其他过程所花费的时间。

最后，将 AMMAS 与文献[10]中的蚁群算法 ACO-OP 相比。由于 ACO-OP 只给出小规模算例的最大总收益(ACO-OP 仅在点数小于 64 的算例上测试)，本节仅比较这些最大总收益。表 5-3 中给出的是 AMMAS 和 ACO-OP 获得的新的最大总收益，它们在其他算例中的结果相同。这表明 AMMAS 是一类有效求解定向问题的蚁群算法。

表 5-3 AMMAS 与 CGW、ACO-OP 的最大总收益

算　例	CGW	ACO-OP	AMMAS
21×30	265	265	**275**
21×40	395	395	**400**
32×60	220	**225**	**225**

5.6 小结

本章应用自适应最大最小蚂蚁系统求解定向问题。证明了差异量在定向问题中是一个距离，分析了差异量的性质，提出了一个快速计算平均差异量的算法。本章还考虑了具有信息素重新初始化机制的最大最小蚂蚁系统。实验表明，与 MMAS 和具有信息素重新初始化机制的 MMAS 相比，自适应最大最小蚂蚁系统的计算结果较好。这一方面表明自适应选取信息素能有效改善算法性能，另一方面也表明，保留蚂蚁在以前积累的信息对算法性能是有利的。与文献中的其他启发算法相比，本章所提出的算法 AMMAS 能在较短的时间内获得更好的解。

参考文献

[1] Golden B L, Levy L, Vohra R. The orienteering problem[J]. Naval Research Logistics, 1987, 34: 307～318.

[2] Hayes M, Norman J M. Dynamic Programming in Orienteering: Route Choice and

the Siting of Controls[J]. Journal of the Operational Research Society，1984，35(9)：791～796.
[3] Laporte G，Martello S. The Selective Traveling Salesman Problem[J]. Discrete Applied Mathematics，1990，26：193～207.
[4] Leifer A C，Rosenwein M S. Strong linear programming relaxations for the orienteering problem[J]. European Journal of Operational Research，1994，73：517～523.
[5] Fischetti M，Gonzales J S，Toth P. Solving the orienteering problem through branch-and-cut[J]. INFORMS Journal of Computer，1998，10：133～148.
[6] Tsiligirides T. Heuristic methods applied to orienteering[J]. Journal of Operations Research Society，1984，35：797～809.
[7] Ramesh R，Brown K M. An efficient four-phase heuristic for the generalized orienteering problem [J]. Computers and Operations Research，1991，18：151～165.
[8] Chao I M，Golden B L，Wasil E A. A fast and effective heuristic for the orienteering problem[J]. European Journal of Operational Research，1996，88：475～489.
[9] Wang Q，Sun X，Golden B L，et al. Using artificial neural networks to solve the orienteering problem[J]. Annals of Operations Research，1995，61：111～120.
[10] Liang Y C，Smith A E. An ant colony approach to the orienteering problem[J]. Journal of the Chinese Institute of Industrial Engineers，2006，23(5)：403～414.
[11] Stützle T，Hoos H H. MAX-MIN ant system[J]. Future Generation Computer Systems，2000，16(8)：889～914.



团队定向问题

6.1 问题描述

在团队定向问题(team orienteering problem, TOP)中,车队中的每个车辆试图访问一组具有一定收益的点。每个车辆必须在规定的时间内从起点出发到达终点。一旦某个车辆经过一个点,它将获得该点对应的收益,其他车辆即使经过该点也不能获得该点对应的收益。团队定向问题的优化目标是使得车队的总收益最大化。

1994 年,Butt 和 Cavalier 最先研究团队定向问题,不过他们称这类问题为多路最大收集问题(Multiple Tour Maximum Collection Problem)[1]。团队定向问题这个名字是由 Chao 等[2]于 1996 年确立的。

许多实际问题可以归结为团队定向问题。例如某电脑公司维修员要给分布在城市各地的顾客提供服务,一旦他们完成了维修任务,他们将获得一定的报酬(收益)。现在要求在一个工作日中安排维修员的行程使公司的收益最大化。在这个问题中,维修员对应于团队定向问题中的车辆,时间限制为一个工作日。由于存在时间限制,维修员可能只能给部分顾客提供服务,因此在调度的时候必须考虑所有顾客分布情况。团队定向问题还可建模其他实际问题,例如多车辆家用燃料配送问题[3]、大学足球队员招募问题[1]、团队定向运动[2]、一些涉及公共承运人和私有运营商的装载和配送服

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

$$\sum_{i=1}^{n-1}\sum_{j>i}c_{ij}x_{ijk} \leqslant T_{\max}(k=1,2,\cdots,m) \tag{6-5}$$

$$\sum_{\substack{i,j\in U\\ i<j}}x_{ijk} \leqslant |U|-1(U\subset V\backslash\{1,n\};\quad 2\leqslant|U|\leqslant n-2;k=1,2,\cdots,m) \tag{6-6}$$

$$x_{ijk}\in\{0,1\},\quad (1\leqslant i<j\leqslant n;k=1,2,\cdots,m) \tag{6-7}$$

$$y_{1k}=y_{nk}=1,\ y_{ik}\in\{0,1\}(i=2,\cdots,n-1;k=1,2,\cdots,m) \tag{6-8}$$

其中约束(6-2)限定每辆车必须从起点 1 出发终止于点 n；约束(6-3)规定路径的连通性；约束(6-4)规定除了点 1 和点 n 之外其他点最多经过一次；约束(6-5)描述了时间约束；约束(6-6)避免出现子路径(即任一条边最多包含于一条路径中)；约束(6-7)和(6-8)限定了变量必须取值为 0 或 1。

6.2 现有算法回顾

由于团队定向问题是 NP-hard 问题，现有研究主要集中于启发算法。Butt 和 Cavalier[1] 提出了一个贪婪算法。Chao 等[2]提出了五步法(初始化、主移动、整理、局部改进、再初始化)，他们还在 Tsiligirides[8] 随机算法的基础上提出了一种启发算法。Tang 和 Miller-Hooks[7] 提出了一种禁忌算法。Archetti 等[9]提出了两种禁忌算法和一种变邻域算法。针对该问题，研究者也提出了两个精确算法(exact algorithm)：一个是基于列生成(column generation based)的精确算法[10]，另一个是基于分支定价(branch and price based)的精确算法[11]。但是它们只能在合理的时间内求解很小规模的问题。例如，在两小时内，后一个算法只能求解测试数据集中的少数几个 100 维的问题。

6.3 算法描述

在应用蚁群算法[12]求解团队定向问题时，解的构造是一个关键点。在许多应用中，解的构造过程是自然的。例如应用蚁群算法求解 TSP 时，在每个构造步，只需要从未被选取的点中按照状态转移规则选取一个点即可。但是对于团队定向问题，在每个构造步，蚂蚁必须决定哪个车辆移动以及决定该车辆向哪个点移动。为解决这一问题，提出了 4 种构造法。

本章所提出的算法(ACO-TOP)在经典蚁群算法框架下求解团队定向问题，其基本流程见图 6-1。下面将详细讨论该算法的特点。首先给出了

信息素和启发信息的定义；接着介绍解的构造和信息素的更新规则；最后介绍局部搜索。

```
初始化信息素并且设置其他参数；
本次迭代的最优解 s_ib ←Null；
当前最优解 s_gb ←Null；
N_ni ← 0；
迭代次数 iteration ←0；
while iteration 小于最大迭代数 N_C do
    for  i = 1 to n_a do (n_a 是蚂蚁数)
        s_i ← ConstructSolution(τ,η)；
        s_i ← LocalSearch(s_i)；
    end for
    s_ib ←argmax(F(s_1),F(s_2),…,F(s_{n_a}))；
    if F(s_ib)>F(s_gb)
        s_gb ←s_ib；
        N_ni ← 0；
    else
        N_ni ← N_ni + 1；
    end if
    PheromoneUpdate(τ,s_ib,s_gb,N_ni)；
    iteration ← iteration + 1；
end while
输出 s_gb
```

图 6-1　ACO-TOP 的伪代码

6.3.1　信息素和启发信息的定义

根据团队定向问题的定义，它可以表示为一个构造图(construction graph)，其顶点就是原问题的顶点。任意边(i,j)都赋有一定量的信息素$\tau(i,j)$，用于表征从一个顶点转移到下一个顶点的偏好(desirablity)。

假设蚂蚁当前位置为i，启发信息$\eta(i,j)$用来表征一种先验的偏好。注意到团队定向问题的优化目标是在规定时间内，使得团队的总收益最大化，因此蚂蚁更加偏好那些具有较高收益且离i较近的点。

这一观察启发我们考虑点j的以下性质：①收益r_j；②i和j之间的距离c_{ij}；③如果所有点都位于同一平面且任意两点直线可达(文献[2]中的算例满足这个条件)，此时可以考虑$\angle jin$的度数，即$\arccos w_{ij}$，其中$w_{ij}=(c_{ij}^2+c_{in}^2-c_{jn}^2)/2c_{ij}c_{in}$。在图 6-2 中，给出了$\angle jin$的示意图。点$j$和$j^*$对应

的角度分别为θ和θ^*，且$\theta<\theta^*$。由图6-2可见，如果选取j^*，则使车辆偏离点n；而如果选取j，则使车辆偏向于点n；考虑到时间约束，点j(从局部来看)优于点j^*。依据这些性质，启发信息的定义如下：

$$\eta(i,j)=\frac{r_j}{c_{ij}}\exp(\gamma w_{ij}) \tag{6-9}$$

其中γ是一参数($\gamma\geqslant 0$)，它决定w_{ij}的重要性。当$\gamma=0$，此时不考虑w_{ij}，且$\eta(i,j)$与Tsiligirides[8]提出的启发信息一致。

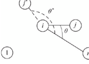


图6-2　$\angle jin$的示意图

6.3.2　解的构造

由约束(6-5)可知，违反式(6-10)的点一定是不可行点[2]。

$$c_{i1}+c_{in}\leqslant T_{\max}(2\leqslant i\leqslant n-1) \tag{6-10}$$

如果事先去掉这些不可行点，就可以减少需要考虑的点数。为方便起见，不失一般性，假定所有的点都满足式(6-10)。

约束的处理是构造解的一个关键点。一种可行方法是将原问题转化为无约束问题，常见的方法有罚函数法(包括内点法、外点法)。但是罚函数法中，罚因子的选取是个难点。蚁群算法是一类构造性算法，它能在每个构造步去掉不可行点，这为处理约束提供了方便。

在构造解的过程中，蚂蚁相当于一个具有简单智能的决策者。它负责为每个车辆选取一条可行路径(即在规定时间内从起点出发到达终点的路径)。具体而言，在每个构造步，蚂蚁选取一车辆并为该车辆选取一个可行点，直到所有的车辆都到达终点(这一过程对应于图6-1中的ConstructSolution过程)。本章考虑以下方法：

(1) 串行法。在这种方法中，蚂蚁先给一个车辆选择一条可行路径，然后再给下一个车辆选择一条可行路径，直到所有车辆都安排了一条可行路径。

(2) 同步法。在这种方法中，蚂蚁按照一定次序给每个车辆选择一可行点，这一过程反复进行，直到所有车辆都安排了一条可行路径。考虑两种

安排车辆的次序：一种是确定性次序，即给车辆选择点的次序是固定不变的；另一种是随机性次序，即给车辆选择点的次序是随机的。相应地，这两种次序对应的方法分别称为确定性同步法和随机性同步法。需要注意的是，如果某个车辆没有可行点，则在以后的构造过程中不再考虑。

(3) 同时法。在这种方法中，蚂蚁从连接每个车辆当前所在点到所有可行点的边中选择一个边。该方法同时确定一个车辆和一个可行点。

考虑一个总点数 n 为 7，且车辆数 m 为 2 的团队定向问题。图 6-3 到图 6-6 分别给出了 4 种构造方法的示意效果。在每个图中，边上的数字表示构造步的序号。图 6-3 显示的是串行法：从第一步到第三步，蚂蚁给第一个车辆安排可行点，而在其他步中给第二个车辆安排可行点。图 6-4 和图 6-5 分别显示的是确定性同步法和随机性同步法。这两种方法交替地给两个车辆安排可行点。在图 6-6 中显示的是同时法，在每一步中蚂蚁同时确定一个车辆和一个点。

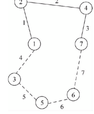


图 6-3 串行法示意图

点数为 7，车辆数为 2，实线是车辆 1 的路径，虚线是车辆 2 的路径。边上的数字表示构造的次序。

为了在蚁群算法的框架下描述这 4 种方法，采用以下记号：

u_i：在第 k 个构造步，第 i 个车辆所处的点（$1 \leqslant i \leqslant m$）。

C_{u_i}：由所有未被经过且满足式(6-11)的点组成的集合。即

$$\forall v \in C_{u_i}, \quad L(t_i) + c_{u_i v} + c_{vn} \leqslant T_{\max} \tag{6-11}$$

其中 $L(t_i)$ 是第 i 个车辆经过的(未完成)路径 t_i 的长度。如果 C_{u_i} 是空集，即没有可行点，则终点被选取并且第 i 条路径完成。

v_k：在第 k 构造步选取的点。

q_k：在第 k 构造步选取的车辆。

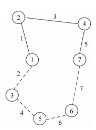


图 6-4　确定性同步法示意图

点数为7,车辆数为2,实线是车辆1的路径,虚线是车辆2的路径。边上的数字表示构造的次序。

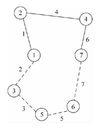


图 6-5　随机性同步法示意图

点数为7,车辆数为2,实线是车辆1的路径,虚线是车辆2的路径。边上的数字表示构造的次序。

在串行法、确定性同步法和随机性同步法中,蚂蚁按照以下概率选择下一点:

$$p(v_{k+1}=v,q_{k+1}=j \mid C_{u_i},1\leqslant i\leqslant m,q_k,\tau)$$

$$=\begin{cases}\dfrac{\tau(u_j,v)^{\alpha}\cdot\eta(u_j,v)^{\beta}}{\sum\limits_{w\in C_{u_j}}\tau(u_j,w)^{\alpha}\cdot\eta(u_j,w)^{\beta}}, & v\in C_{u_j}\\ 0, & \text{其他}\end{cases}\tag{6-12}$$

其中,α 和 β 是参数。注意到在串行法中,车辆 q_k 和 q_{k+1} 是相同的,而在两

种同步法中它们可能是不同的。虽然这个概率决策规则与基本蚁群算法的随机概率规则形式上有所不同，但是，它在选取下一个移动方向时都偏好于选取启发信息和信息素较大的边，其基本思想与基本蚁群算法的随机概率规则类似。

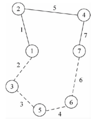


图 6-6 同时法示意图

点数为 7，车辆数为 2，实线是车辆 1 的路径，虚线是车辆 2 的路径。边上的数字表示构造的次序。

在同时法中，蚂蚁按照式(6-13)给出的概率选择下一个点：

$$p(v_{k+1}=v, q_{k+1}=j \mid C_{u_i}, 1\leqslant i\leqslant m, q_k, \tau)$$
$$=\begin{cases}\dfrac{\tau(u_j,v)^{\alpha}\cdot\eta(u_j,v)^{\beta}}{\sum\limits_{i=1}^{m}\sum\limits_{w\in C_{u_i}}\tau(u_i,w)^{\alpha}\cdot\eta(u_i,w)^{\beta}}, & v\in C_{u_j}\\ 0, & \text{其他}\end{cases} \tag{6-13}$$

由于在每个构造步只有一个车辆和一个点被选取，式(6-13)可以通过保存 $\sum\limits_{w\in C_{u_i}}\tau(u_i,w)^{\alpha}\cdot\eta(u_i,w)^{\beta}(1\leqslant i\leqslant m)$ 来快速地计算。假定车辆 j 和点 v 被选取，则对于车辆 $i(i\neq j)$，它所对应的可行点集变为 $C_{u_i}\setminus\{v\}$。

6.3.3 信息素的更新规则

一旦所有的蚂蚁都构造了一个解，按照最大最小蚂蚁系统给出的规则更新信息素(即图 6-1 中 PheromoneUpdate)：

$$\tau(u,v)^{t+1}=\rho\tau(u,v)^{t}+\Delta\tau(u,v) \tag{6-14}$$

$$\text{如果 } \tau(u,v)^{t+1}<\tau_{\min},\quad \text{则 } \tau(u,v)^{t+1}=\tau_{\min} \tag{6-15}$$

$$\text{如果 } \tau(u,v)^{t+1}>\tau_{\max},\quad \text{则 } \tau(u,v)^{t+1}=\tau_{\max} \tag{6-16}$$

其中 $\tau(u,v)^l$ 是边 (u,v) 在第 l 代时的信息素。如果 (u,v) 在第 l 代被最优蚂蚁经过，则 $\Delta\tau(u,v)$ 等于 $F(s_{\text{best}})$，否则，$\Delta\tau(u,v)=0$。最优解 s_{best} 可能是本次迭代获得的最优解 s_{ib} 也可能是当前最优解 s_{gb}。$F(x)$ 是质量函数(quality function)[13]，它由下式给出：

$$F(x)=\sum_{i=2}^{n-1}\sum_{k=1}^{m}r_i y_{ik}\Big/\sum_{i=2}^{n-1}r_i \tag{6-17}$$

$\tau_{\max}$ 和 $\tau_{\min}$ 分别是信息素的上下界。设置上下界的目的是为了避免停滞。它们按照式(6-18)和式(6-19)设置为

$$\tau_{\max}=\frac{F(s_{gb})}{(1-\rho)} \tag{6-18}$$

$$\tau_{\min}=(1-\sqrt[n]{P_{\text{best}}})/((avg-1)\sqrt[n]{P_{\text{best}}})\tau_{\max} \tag{6-19}$$

其中 avg 等于 $n/2$，P_{best} 是参数 $(0\leqslant P_{\text{best}}\leqslant 1)$。依据 Stützle 和 Hoos 的建议，信息素初始化为任意大的数值，这使得算法在运行之初有较好的探索能力。另外，重新初始化信息素能起到改善算法性能的作用。在算法运行过程中，如果从找到上一个当前最优解算起经过 N_{ni} 代仍然没有找到新的最优解，所有信息素将被设置为信息素的上界。

6.3.4 局部搜索

局部搜索常用来改善蚁群算法的性能，但是在应用中要折中考虑解质量和计算时间(即图 6-1 中的 LocalSearch 过程)。虽然目前已有一些有效的局部搜索，本章利用了文献[2]中的局部搜索。其基本思想是：先用 2-opt 法缩短解中每条路径的长度，再插入尽可能多的可行点。这个过程反复进行，直到路径长度不能缩短或者不能再插入可行点时终止。在算法中，局部搜索对每个构造解进行改进，它在所有蚂蚁都构造完解之后且在信息素还未更新之前执行。

6.4 实验分析

本节用实验来分析文中所提出的算法并比较 4 种构造法。所有代码用 C++ 实现，运行环境为：3GHz CPU，1G RAM，Windows XP 操作系统。采用文献[2]中的 387 个测试算例，它们分别属于 7 个数据集。这些数据集的顶点数目分别为 32、21、33、100、66、64、102。在每个数据集中，每个点的坐标和对应的收益不同。每个数据集由 3 组数据子集构成，每个数据子集的车辆数分别为 2、3、4。在数据子集中，每个算例的区别仅在于 $T_{\max}$。每个数据子集用 $x.y$ 表示，其中 x 表示该数据子集所在数据集的序号，y 表示车辆

数。每个算例用 $x.y.z$ 表示,其中 x 表示该算例所在数据集中的序号,y 表示车辆数,z 表示该算例在相应数据子集中的序号。

ACO-TOP 算法用混合策略更新信息素:每隔 5 代,当前最优解 s_{gb} 被用来更新信息素,在其他代中,都用本次迭代获得的最优解 s_{ib} 来更新信息素。对于每个算例,测试 10 次。

6.4.1 参数设置

在测试算法之前,首先确定算法的参数。考虑到计算时间与计算结果,最大运行代数 N_C 为 2000,蚂蚁数为 20。$N_{ni}=250$。如蚁群算法在许多应用中一样,$\alpha=1$,$\rho=0.98$,$P_{\text{best}}=0.05$。实验表明,参数 β 和 γ 对算法性能起着关键作用。我们比较了不同的 β 和 γ,它们的测试值分别为:$\beta\in\{0, 0.25, 0.5, 1, 2, 4, 8\}$,$\gamma\in\{0, 0.25, 0.5, 1, 2, 4, 8\}$。数值结果表明当两者都取为 0.5 时,算法性能较好。

6.4.2 4 种构造法的比较

在表 6-1 和表 6-2 分别给出了 4 种构造方法在两个规模最大的数据集(即第 4 个和第 7 个数据集)中的计算结果。由于在其他数据集中 4 种构造方法的差异不大,详细计算结果在附录中给出。由结果可见,串行法和同时法最好。确定性同步法与随机性同步法的计算结果相当,前者略好。在大多数算例中,串行法的最大值优于其他 3 种方法。在第 4 个数据集中,同时法的平均值较好,而在第 7 个数据集中,串行法的平均值较好。

表 6-1 4 种构造方法在第 4 个数据集中的实验结果

算 例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p4.2.a	206	206	206	206	206	206	206	206
p4.2.b	341	338.7	341	338	341	338.7	341	340.2
p4.2.c	452	447.9	452	448.8	452	449.4	452	448
p4.2.d	531	527.5	531	528.7	530	528.4	531	528.2
p4.2.e	618	596.9	600	595.6	600	597	613	599.5
p4.2.f	687	672.6	672	667.6	672	663.8	672	664.9
p4.2.g	757	736.8	756	743.2	756	746.1	756	749.4
p4.2.h	827	818.2	819	812.8	819	812.4	820	815
p4.2.i	918	894.1	900	888.5	918	873.8	918	895.3
p4.2.j	965	953.2	962	949.6	962	945.2	962	960.2
p4.2.k	1022	1001.1	1016	1001.3	1016	1004.2	1016	1001.8

续表

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p4.2.l	1071	1063.5	1070	1062.2	1071	1058.9	1069	1060.8
p4.2.m	1130	1110.6	1115	1106.9	1119	1108.3	1113	1094.2
p4.2.n	1168	1146.9	1149	1133.6	1158	1148	1169	1146.6
p4.2.o	1215	1175.8	1209	1188	1198	1184.3	1210	1184.1
p4.2.p	1242	1215	1229	1211.7	1233	1206.9	1239	1206.3
p4.2.q	1263	1234.3	1253	1232.6	1252	1225.7	1260	1227.2
p4.2.r	1288	1263.4	1278	1257.5	1278	1261.6	1279	1264
p4.2.s	1304	1288.4	1304	1288.4	1303	1284.9	1304	1294.5
p4.2.t	1306	1304.4	1306	1305.1	1306	1303	1306	1306
p4.3.b	38	38	38	38	38	38	38	38
p4.3.c	193	193	193	193	193	193	193	193
p4.3.d	335	333	333	332.5	333	333	335	332
p4.3.e	468	463.2	468	465.6	468	466.4	468	465.6
p4.3.f	579	569.2	579	575.6	579	573.8	579	569
p4.3.g	653	651.6	652	652	653	647.2	652	649.4
p4.3.h	720	712.6	713	709.8	713	709.4	713	710.4
p4.3.i	796	779.2	793	778	793	781.9	786	775.6
p4.3.j	861	839.4	857	845.6	855	841.4	858	850.5
p4.3.k	918	895.7	913	900.7	910	899	910	896.6
p4.3.l	979	954.2	958	952.4	976	961.1	966	953.4
p4.3.m	1053	1023.1	1039	1019.8	1028	1003.4	1046	1028.8
p4.3.n	1121	1100.3	1109	1093.9	1112	1099.7	1103	1094.7
p4.3.o	1170	1158.1	1163	1154.2	1167	1155.6	1165	1157.6
p4.3.p	1221	1201.7	1202	1189.4	1207	1200.8	1207	1202.2
p4.3.q	1252	1227.4	1239	1232.8	1239	1221.8	1238	1231
p4.3.r	1267	1255.7	1263	1260.4	1263	1260.4	1263	1260.2
p4.3.s	1293	1283.7	1291	1284.9	1289	1282	1291	1286.2
p4.3.t	1305	1302.3	1304	1302.8	1303	1293.6	1304	1301.8
p4.4.d	38	38	38	38	38	38	38	38
p4.4.e	183	183	183	183	183	183	183	183
p4.4.f	324	324	324	323.5	324	322.2	324	323.5
p4.4.g	461	460.1	461	459.8	461	458.3	460	460
p4.4.h	571	552	556	556	556	555.2	556	554.2
p4.4.i	657	641.6	653	642.6	652	643.6	653	649.1
p4.4.j	732	726.7	731	721.2	711	707.3	731	726.8
p4.4.k	821	814.2	820	815.3	818	813	818	814

续表

算例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p4.4.l	880	868.4	877	871.5	875	870.2	875	870.3
p4.4.m	918	904.7	911	909.1	906	903.1	911	906.9
p4.4.n	961	946.3	956	948.9	956	948	956	952.3
p4.4.o	1036	1001.1	1030	1012.3	1021	1002.7	1029	1015.5
p4.4.p	1111	1074	1108	1073.5	1088	1064.4	1110	1099.4
p4.4.q	1145	1106.2	1150	1117.2	1137	1107.7	1148	1122.5
p4.4.r	1200	1168.7	1195	1153	1195	1163.2	1194	1161.2
p4.4.s	1249	1233.9	1256	1229.2	1249	1213.7	1252	1238.1
p4.4.t	1281	1268.4	1281	1276.2	1283	1273.5	1281	1268.6

表 6-2　4 种构造方法在第 7 个数据集中的实验结果

算例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p7.2.a	30	30	30	30	30	30	30	30
p7.2.b	64	64	64	64	64	64	64	64
p7.2.c	101	101	101	101	101	101	101	101
p7.2.d	190	190	190	190	190	190	190	190
p7.2.e	290	290	290	290	290	290	290	290
p7.2.f	387	386.7	387	386.7	387	387	387	386.4
p7.2.g	459	459	459	459	459	459	459	459
p7.2.h	521	521	521	520.6	521	521	521	521
p7.2.i	580	578.6	579	578.3	579	578.3	579	578.3
p7.2.j	646	644	646	644.6	646	645.7	646	644.3
p7.2.k	705	701.2	704	701.8	704	702.8	704	702.8
p7.2.l	767	765.4	767	765.5	767	766.5	767	764.3
p7.2.m	827	827	827	824.5	827	825.8	827	826.4
p7.2.n	888	878	878	878	878	878	878	877.4
p7.2.o	945	940.1	945	935.8	940	933.2	941	935.1
p7.2.p	1002	991.3	991	983.5	993	985.4	993	986.6
p7.2.q	1043	1040	1042	1038.6	1043	1036.5	1043	1033.4
p7.2.r	1094	1078.9	1093	1082.9	1088	1075.6	1094	1084.4
p7.2.s	1136	1115.2	1136	1118.3	1134	1119.2	1131	1115.7
p7.2.t	1179	1146.6	1179	1161.9	1179	1153.1	1179	1157.1
p7.3.b	46	46	46	46	46	46	46	46
p7.3.c	79	79	79	79	79	79	79	79

续表

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p7.3.d	117	117	117	117	117	117	117	117
p7.3.e	175	175	175	175	175	175	175	175
p7.3.f	247	247	247	247	247	247	247	247
p7.3.g	344	344	344	344	344	344	344	344
p7.3.h	425	424.3	425	423.9	425	423.1	425	424.5
p7.3.i	487	485.3	487	485.1	486	485.6	487	485
p7.3.j	564	563.2	564	562.8	564	563.4	564	563.3
p7.3.k	633	629.5	632	627.1	633	629.4	633	629.6
p7.3.l	684	680.7	683	680.5	684	679	684	681.2
p7.3.m	762	759.1	762	756.3	762	754.2	762	755.5
p7.3.n	820	813.9	819	811	819	811.2	820	813
p7.3.o	874	874	874	873.7	874	873	874	873.7
p7.3.p	929	925.6	925	922.3	926	924.1	925	923.6
p7.3.q	987	984.5	987	983.1	987	982.5	987	981
p7.3.r	1026	1018.4	1024	1017	1021	1015	1022	1016.4
p7.3.s	1081	1070.3	1081	1062.2	1081	1062.6	1077	1061.5
p7.3.t	1118	1107.2	1117	1101	1103	1086.5	1117	1108
p7.4.b	30	30	30	30	30	30	30	30
p7.4.c	46	46	46	46	46	46	46	46
p7.4.d	79	79	79	79	79	79	79	79
p7.4.e	123	123	123	123	123	123	123	123
p7.4.f	164	164	164	164	164	164	164	164
p7.4.g	217	217	217	217	217	217	217	217
p7.4.h	285	285	285	285	285	285	285	285
p7.4.i	366	366	366	366	366	366	366	366
p7.4.j	462	462	462	461.7	462	461.1	462	462
p7.4.k	520	518	520	517.2	520	517.8	520	517.9
p7.4.l	590	581.7	590	580.5	590	583.6	590	584.8
p7.4.m	646	643.9	644	642.9	646	643.4	646	642.2
p7.4.n	730	725.6	725	724.4	725	724.4	726	724.5
p7.4.o	781	777.5	778	775.2	781	776.2	778	776.5
p7.4.p	846	839.4	846	838.7	838	832.9	842	835.5
p7.4.q	909	905.1	909	905.6	909	904.1	909	904.2
p7.4.r	970	969.2	970	968.8	970	968.4	970	966.5
p7.4.s	1022	1017.7	1019	1014.8	1021	1014.5	1019	1013.4
p7.4.t	1077	1072.8	1072	1070.5	1077	1071.1	1077	1071.5

表 6-3 给出了 21 组数据子集的实验结果。对每一种方法，表中给出了每一组的最大值、平均值。由表 6-3 的结果可见，在数据集 1 中，4 种方法的计算结果完全相同，且在每一次测试中都找到最好解。在其他数据集中，结果再次表明，串行法获得的结果最好，同时法次之。另外实验发现，串行法需要的计算时间较大，但是，它能在 51.1s 内（平均意义下）求解每个算例。因此，串行法能在解质量（solution quality）和计算时间之间较好地折中。

表 6-3 4 种构造方法在 21 个数据子集中的平均计算结果

数据子集	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
1.2	**149.1**	**149.1**	**149.1**	**149.1**	**149.1**	**149.1**	**149.1**	**149.1**
1.3	**125.0**	**125.0**	**125.0**	**125.0**	**125.0**	**125.0**	**125.0**	**125.0**
1.4	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**
2.2	**190.5**	**190.5**	**190.5**	**190.5**	**190.5**	**190.5**	**190.5**	**190.5**
2.3	**136.4**	**136.4**	**136.4**	**136.4**	**136.4**	**136.4**	**136.4**	**136.4**
2.4	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**
3.2	**496.0**	**496.0**	**496.0**	495.8	**496.0**	495.8	**496.0**	**496.0**
3.3	**411.5**	**411.4**	**411.5**	411.2	**411.5**	411.1	**411.5**	411.2
3.4	**336.5**	**336.5**	**336.5**	**336.5**	**336.5**	336.2	**336.5**	336.3
4.2	**915.6**	**899.8**	908.4	898.3	909.5	897.3	911.8	899.6
4.3	**853.8**	841.1	847.7	841.1	848.4	840.1	848.2	**841.9**
4.4	**798.1**	783.0	795.9	784.1	791.4	780.4	795.2	**787.3**
5.2	**897.6**	**891.1**	896.4	890.5	896.2	892.2	896.2	890.7
5.3	**782.8**	**775.3**	780.4	774.6	781.2	774.5	781.0	774.2
5.4	**708.8**	**704.6**	707.7	698.5	706.3	698.2	705.6	698.2
6.2	**819.3**	815.4	818.7	814.1	818.7	814.8	**819.3**	**815.7**
6.3	**792.8**	786.0	790.5	785.6	790.5	784.7	791.3	**786.3**
6.4	**714.0**	701.3	**714.0**	699.1	**714.0**	699.5	**714.0**	**702.0**
7.2	**642.7**	637.4	641.5	**637.7**	641.0	637.1	641.2	637.4
7.3	**599.9**	**597.1**	599.4	595.5	598.6	594.6	599.2	596.0
7.4	**519.1**	**517.0**	518.2	516.3	518.4	516.2	518.4	516.3

6.4.3 与其他算法的比较

本章所提出的算法与下列算法进行了比较：

CGW：Chao 等[2]提出的五步法；

TMH：Tang 和 Miller-Hooks[7]提出的禁忌算法；

GTP：Archetti 等[9]提出的基于罚方法的禁忌算法；

GTF：Archetti 等提出的基于可行方法的禁忌算法；

FVF：Archetti 等提出的快速变邻域算法；

SVF：Archetti 等提出的慢变邻域算法，它与 FVF 的主要区别在于参数的选取。

由于上述文献中只给出了相应算法求得的每个算例的最大收益(最大值)，本章只比较最大收益。在数值实验时，路径长度的精度是一个必须考虑的问题。在 CGW 中，Chao 等没有采用实数精度，而是把路径长度四舍五入到小数点后第一位。而其他文献没有明确指出它们的数值精度。显然，当采用实数精度时所得到的结果可能较差。注意到当路径长度四舍五入到小数点后第一位时，ACO-TOP(采用串行法构造问题的解)能得到 19 个算例的新的最大总收益，结果在表 6-4 中给出。为了与现有算法相比较，ACO-TOP 主要考虑实数精度。

表 6-4　ACO-TOP 求得的新的最大总收益(路径长度四舍五入到小数点后第一位)

算　例	n	m	$T_{\max}$	ACO-TOP
p1.2.e	32	2	12.5	50
p4.2.b	100	2	30.0	344
p4.2.k			75.0	1023
p4.3.c	100	3	23.3	194
p4.3.d			26.7	336
p5.3.f	66	3	10.0	135
p5.3.q			28.3	1080
p5.3.r			30.0	1145
p5.3.s			31.7	1225
p5.3.y			41.7	1600
p5.4.m	66	4	16.2	590
p5.4.p			20.0	780
p5.4.x			30.0	1500
p6.2.e	64	2	17.5	384
p6.2.j			30.0	972
p6.3.h	64	3	16.7	462
p6.4.k	64	4	16.2	558
p7.2.g	102	2	70.0	467
p7.4.q	102	4	85.0	912

对于每个算法，每个数据子集的(平均)最大总收益在表 6-5 中给出。可见 ACO-TOP 和 SVF 的计算结果最好，GTF 和 FVF 次之，GTP 比以上

算法结果略差，而 CGW 和 TMH 的结果最差。ACO-TOP 和 SVF 分别在 15 个数据子集中获得优于其他算法的结果。另外，ACO-TOP 能找到 347 个算例已知最大收益，并且它能找到 12 个算例新的最大总收益(详细结果在表 6-6 中给出)。由以上比较结果可见，ACO-TOP 具有较好的性能。

表 6-5 7 种算法的计算结果

数据子集	ACO-TOP	CGW	TMH	GTP	GTF	FVF	SVF
1.2	**149.1**	148.5	148.8	**149.1**	**149.1**	**149.1**	**149.1**
1.3	125.0	**125.6**	124.7	125.0	125.0	125.0	125.0
1.4	**101.0**	99.3	**101.0**	**101.0**	**101.0**	**101.0**	**101.0**
2.2	**190.5**	190.0	190.0	**190.5**	**190.5**	**190.5**	**190.5**
2.3	**136.4**	135.9	135.9	**136.4**	**136.4**	**136.4**	**136.4**
2.4	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**	**94.5**
3.2	**496.0**	488.5	492.0	494.5	**496.0**	**496.0**	**496.0**
3.3	**411.5**	403.0	408.0	**411.5**	**411.5**	**411.5**	**411.5**
3.4	**336.5**	332.5	335.0	**336.5**	**336.5**	**336.5**	**336.5**
4.2	915.6	875.7	895.1	904.9	908.5	914.0	**916.2**
4.3	853.8	815.1	844.3	845.5	852.5	853.0	**855.6**
4.4	798.1	766.1	784.6	800.1	802.3	801.7	**803.2**
5.2	**897.6**	890.6	886.8	892.6	897.4	895.8	897.0
5.3	783.4	776.6	775.8	781.4	**783.6**	**783.6**	**783.6**
5.4	**708.8**	696.0	699.0	707.5	**708.8**	**708.8**	**708.8**
6.2	**819.3**	814.9	818.2	813.8	818.7	**819.3**	**819.3**
6.3	**792.8**	787.5	783.0	**792.8**	**792.8**	**792.8**	**792.8**
6.4	714.0	**716.4**	712.8	714.0	714.0	714.0	714.0
7.2	**642.7**	633.9	633.5	639.6	641.4	640.6	642.5
7.3	**599.9**	585.5	592.5	596.7	597.7	597.1	599.3
7.4	**519.1**	497.4	514.6	517.2	516.9	516.9	518.9

表 6-6 ACO-TOP 求得的新的最大总收益(路径长度为实数精度)

算　例	n	m	$T_{\max}$	ACO-TOP	已知最优
p4.2.j	100	2	70.0	965	962
p4.2.p			100.0	1242	1241
p4.2.r			110.0	1288	1286
p4.2.s			115.0	1304	1301
p4.3.q	100	3	70.0	1252	1251
p4.3.t			80.0	1305	1304
p5.2.y	66	2	62.5	1645	1635

续表

算　　例	n	m	$T_{\max}$	ACO-TOP	已知最优
p7.2.i	102	2	90.0	580	579
p7.2.j			100.0	646	644
p7.3.l	102	3	80.0	684	683
p7.3.p			106.7	929	927
p7.3.t			133.3	1118	1117

表 6-7 给出了每个算法的平均计算时间。注意到 CGW 在 SUN 4/730 Workstation 25MHz 上运行，TMH 在 DEC Alpha XP1000 computer 667MHz 运行，而其他算法在 2.8GHz PC 上运行。这些算法的运行环境不一致，难以直接比较时间。SVF 运行的 PC 性能与 ACO-TOP 运行的 PC 接近，它在两个规模最大的数据集中的运行时间都远远超过 1min。SVF 在第 4 个数据集中的计算时间为 1118s(将近 20min)，而在第 7 个数据集中的计算时间为 911s(将近 15min)。ACO-TOP 在第 4 个数据集中耗时最长(51.1s)，但是它能在 1min 内求解每个问题，因此 ACO-TOP 算法可以在合理的时间内获得令人满意的解。

表 6-7　7 种算法的计算时间(s)

	ACO-TOP	CGW	TMH	GTP	GTF	FVF	SVF
Set 1	7.9	15.4	N.A.	10.0	5.0	1.0	22.0
Set 2	3.8	0.9	N.A.	0.0	0.0	0.0	1.0
Set 3	8.5	15.4	N.A.	10.0	9.0	1.0	19.0
Set 4	51.1	934.8	796.7	612.0	324.0	121.0	1118.0
Set 5	25.2	193.7	71.3	147.0	105.0	30.0	394.0
Set 6	20.3	150.1	45.7	96.0	48.0	20.0	310.0
Set 7	44.7	841.4	432.6	582.0	514.0	90.0	911.0

6.5　小结

针对团队定向问题，提出了求解该问题的蚁群算法(ACO-TOP 算法)。在经典蚁群算法的基本框架下，提出了串行法、确定性同步法、随机性同步法和同时法来构造问题的解。

本章依据在经典算例中获得的计算结果，比较了这 4 种方法。结果表明，使用串行法能获得较高质量的解。最后，与现有的启发式算法(包括五步法、三种禁忌算法、变邻域算法等)相比，ACO-TOP 算法能在合理的时间

内提供令人满意的解。

ACO-TOP 算法的 4 种构造法相互独立,一种可能的推广是采用混合策略来构造问题的解,即每个蚂蚁构造解的方法可能不同。实验表明,局部搜索能有效改善算法性能,研究高效率的局部搜索是一个有趣的方向。在后续的研究中,将把蚁群算法推广到其他类似问题中,例如具有服务时间(service time)的团队定向问题和车辆路径问题[154]。

需要指出的是,ACO-TOP 算法也可以用来求解定向问题,但是由于采用了局部搜索导致它比第 4 章中的算法所用时间更多。如果算例规模不大,第 4 章中的算法在计算时间和计算结果上较好。但是,对于大规模的问题,可以利用本章的局部搜索。

本章给出的是文献[14]中的实验结果。我们也考虑用自适应最大最小蚂蚁系统来求解团队定向问题,实验结果略有改进。由于自适应最大最小蚂蚁系统采用新的有效的平衡多样性和强化性的方法,它为求解团队定向问题提供了可行的候选方案。

参考文献

[1] Butt SE, Cavalier TM. A Heuristic for the multiple path maximum collection problem[J]. Computers and Operations Research, 1994, 21: 101～111.

[2] Chao IM, Golden B, Wasil EA. The team orienteering problem[J]. European Journal of Operational Research, 1996, 88: 464～474.

[3] Golden B, Levy L, Vohra R. The orienteering problem[J]. Naval Research Logistics, 1987, 34: 307～318.

[4] Ballou R, Chowdhury M. MSVS: an extended computer model for transport mode selection[J]. The Logistics and Transportation Review, 1980, 16: 325～338.

[5] Diaby M, Ramesh R. The distribution problem with carrier service: a dual based penalty approach[J]. ORSA Journal on Computing, 1995, 7: 24～35.

[6] Hall R, Racer M. Transportation with common carrier and private fleets: system assignment and shipment frequency optimization[J]. IIE Transactions, 1995, 27: 217～225.

[7] Tang H, Miller-Hooks E. A tabu search heuristic for the team orienteering problem[J]. Computers and Operations Research, 2005, 32: 1379～1407.

[8] Tsiligirides T. Heuristic methods applied to orienteering[J]. Journal of Operations Research Society, 1984, 35: 797～809.

[9] Archetti C, Hertz A, Speranza MG. Metaheuristics for the team orienteering problem[J]. Journal of Heuristics, 2007, 13: 49-76.

[10] Butt S, Ryan D. An optimal solution procedure for the multiple path maximum

collection problem using column generation[J]. Computers and Operations Research, 1999, 26: 427～441.

[11] Boussier S, Feillet D, Gendreau M. An exact algorithm for team orienteering problems[J]. 4OR, 2007, 5(3): 211～230.

[12] Stützle T, Hoos HH. MAX-MIN ant system[J]. Future Generation Computer Systems, 2000, 16(8): 889～914.

[13] Blum C, Dorigo M. The hyper-cube framework for ant colony optimization[J]. IEEE Transaction on Systems Man and Cybernetic-Part B, 2004, 34(2): 1161～1172.

[14] Ke L, Archetti C, Feng Z. Ants can solve the team orienteering problem[J]. Computers and Industrial Engineering, 2008, 54(3): 648～665.



属性约简

7.1 问题描述

在处理高维数据时，特征提取是学习过程必不可少的一步。特征提取研究如何从一个特征集中选择一个子集使得它能足够准确地描述原始特征集。通过删除不相关和冗余特征，特征提取有助于改进学习算法的速度和精度，同时有助于增强所构造模型的可理解性[1]。

在实际应用中，常常要处理模糊、不精确的属性或特征。而粗糙集是一个有效处理不精确、不确定性、模糊的数学工具[2~5]。基于粗糙集的属性约简(attribute reduction in rough set theory)已经成为一类应用广泛的特征提取方法[6~12]。

一个决策表信息系统可以表示为 $I=(U,A)$，其中 U 是论域，它是由有限对象构成的非空集；A 是非空属性集；对于任意 $a\in A$，$a:U\to V_a$；V_a 是属性 a 的值域。$A=C\cup D$，其中 C 是条件属性集，D 是决策属性集。在一个决策表中，决策属性可能不唯一，根据文献[13]中的方法可以把该决策表转化为单一决策属性的决策表。本章只讨论具有单一决策属性的决策表。

对于属性子集 $P\subseteq A$，定义如下的不可分辨关系(不分明关系)$IND(P)$：

$$IND(P)=\{(x,y)\in U^2 \mid \forall a\in P, a(x)=a(y)\} \tag{7-1}$$

如果 $(x,y)\in IND(P)$，则根据属性子集 P，x 和 y 是不可以区分的。

显然不可分辨关系是一个等价关系。由 $IND(P)$ 确定的等价类划分记为 U/P。包含 x 的等价类记为 $[x]_P$。不可分辨关系是粗糙集的数学基础。

决策表信息系统中的一个条件属性 a 对应着一个等价关系(即不分明关系或不可分辨关系),它对论域 U 形成一个划分 $U/\{a\}$。决策表的所有条件属性形成论域一个划分;同时,决策属性也对论域形成一个划分。这两个划分构成了条件属性和决策属性对论域样本分类的知识。

在粗糙集理论中,上下近似是最重要的概念。给定对象集 $X\subseteq U$,X 的 P-上近似集记为 $\bar{P}X$,它是 U 中根据属性子集 P 可能归于集合 X 的对象组成的集合;X 的 P-下近似集记为$\underline{P}X$,它是 U 中根据属性子集 P 一定归于集合 X 的对象组成的集合。上下近似集的定义如下:

$$\bar{P}X = \{x \mid [x]_P \cap X \neq \varnothing\} \tag{7-2}$$

$$\underline{P}X = \{x \mid [x]_P \subseteq X\} \tag{7-3}$$

由上下近似集的定义可知,$\underline{P}X\subseteq\bar{P}X$。若$\underline{P}X=\bar{P}X$,则 X 称为 P 可定义集;否则,X 称为 P 粗糙集。

在数据分析时,一个关键问题是获取属性之间的依赖关系。依据上下近似集,属性依赖度(degree of depencey)定义如下:

定义 1(依赖度)　令 $P,Q\subseteq A$,则依赖度 κ 的定义如下:

$$\kappa = r_p(Q) = \mid POS_p(Q) \mid / \mid U \mid \tag{7-4}$$

其中$|A|$是相应集合的基数。$POS_p(Q)$称为正区域,它的定义如下:

$$POS_p(Q) = \bigcup_{X\in U/Q} \underline{P}X \tag{7-5}$$

依赖度 κ 衡量 Q 和 P 之间的依赖程度。如果 $\kappa=1$,则 Q 完全依赖于 P;如果 $0<\kappa<1$,则 Q(以程度 κ)部分依赖于 P;如果 $\kappa=0$,则 Q 不依赖于 P。

在决策表信息系统中,一条记录代表一个决策规则,如果把所有决策规则罗列出来,就可以得到一个决策规则集合。但是,这样的决策规则集合只是记录了样本的情况,适应性较弱。为了从决策表中提取适应度较大的规则,需要对决策表进行约简,使经过约简处理的决策表中的一条记录就代表一类具有相同规律的样本,这样得到的决策规则具有较高的适应性。在保持决策表决策属性和条件属性之间的依赖关系不发生变化的前提下,对决策表进行约简或简化,称为基于粗糙集的属性约简。它提供了一种 filter-based 方法来获取知识。在依赖度的基础上,约简(reduct)的定义如下:

定义 2(约简)　设 R 是条件属性集 C 的一个子集,如果它满足[5]

$$r_R(D) = r_C(D) \quad 且 \quad \forall B \subset R, r_B(D) < r_R(D) \tag{7-6}$$

则称 R 为 C 的一个约简。

在原始的决策表信息系统中的条件属性并非是同等重要的，甚至其中某些条件属性是冗余的。属性约简删除其中不必要的或不重要的属性。但是，一个决策表的条件属性对决策属性的约简通常不是唯一的，也就是说，一个决策表可能有多个约简。由于约简后属性的数目直接影响着决策规则的繁简和性能，具有最小基数的约简(即最小约简)是应用较广的一类约简。本章仅仅研究这类约简。相应地，其目标函数为

$$\min_{R\in\Theta} |R| \tag{7-7}$$

其中 Θ 是由属性集 C 所有约简构成的集合。

例　设有一决策表(见表 7-1)，其条件属性为 $\{a, b, c, d\}$，决策属性为 e。由约简的定义可知，$\Theta=\{\{a, b\}, \{a, c\}\}$，最小约简为 $\{a, b\}$ 和 $\{a, c\}$。由此可见，最小约简也可能不唯一。

表 7-1　决策表示例

$O\in U$	a	b	c	d	e
0	2	0	2	1	0
1	1	1	1	0	2
2	1	0	0	2	1
3	2	1	0	1	2
4	0	0	2	1	1
5	1	2	0	2	1
6	1	1	1	2	2
7	0	1	1	0	1

7.2　现有算法回顾

众所周知，属性约简是一个 NP-hard 问题[8]。爬山法和元启发算法是两类重要的属性约简算法。爬山法有两种实现方式，一种是从空集或属性核出发依据属性重要性不断增加属性，当所得到的属性子集是一个约简时，算法终止；另一种方式是从条件属性全集出发不断地删除属性，直到不能再删除为止(否则剩余属性子集不是一个约简)。根据属性重要性的不同，常见的爬山法包括基于正区域算法、基于条件信息熵算法和基于可分辨矩阵算法等。不过，爬山法不能保证能求得最小约简[9]。许多研究者考虑利用元启发算法，如遗传算法[10,11]、模拟退火算法[11]、蚁群算法[11]、禁忌算法[12]、粒子群算法[9]，来解决这一问题。实验表明，这些算法常常能获得高质量的解。但是，这些算法用时较长，即使求解一些中等规模的问题，也需

要几千秒[9,11]。

7.3 算法描述

属性约简可以描述为一个完全图 $G=(V,E)$，其中 V 是顶点集，它对应于条件属性集，E 是由所有边组成的集合。属性约简的优化目标就是在图中寻找一条路径，使经过的点构成的集合是一个约简且它的基数最小。

本章提出了 3 种蚁群算法。依据其信息素释放方式，分别称这 3 种算法为边模式蚁群算法(Edge-mode ACO，简称为 Edge-ACO)、团模式蚁群算法(Clique-mode ACO，简称为 Clique-ACO)和点模式蚁群算法(Vertex-mode ACO，简称为 Vertex-ACO)。下面首先介绍属性约简的基本概念，然后给出了本章提出的算法和实验分析，最后是小结。

本章提出的 3 种算法的主要流程为：在每一步，每个蚂蚁构造一个解，然后依据构造的解更新信息素。算法一直迭代到某个停止条件满足为止。本章采用的停止条件为最大迭代次数。

7.3.1 边模式蚁群算法

在边模式中，每一个边都赋有一定的信息素和启发信息。本章考虑基于依赖度的启发信息。如果某个边的启发信息过小，则蚂蚁经过它的概率就可能过小。为避免这种情形，有必要设置启发信息的取值范围。$\forall a,b\in C$，启发信息 $\eta(a,b)$ 定义如下：

$$\eta(a,b)=|POS_{\{a,b\}}(D)|/|U| \tag{7-8}$$

$$\text{如果 } \eta(a,b)<\varepsilon,\quad \text{则 } \eta(a,b)=\varepsilon \tag{7-9}$$

其中 ε $(0<\varepsilon<1)$ 是一个正参数。

在构造解时，蚂蚁从一个随机选取的属性出发，它依据式(7-10)给出的概率选取下一个属性：

$$p(c_{k+1}=v\mid\tau,c_k=u)=\begin{cases}\dfrac{\tau(u,v)^{\alpha}\cdot\eta(u,v)^{\beta}}{\sum\limits_{w\in C_u}\tau(u,w)^{\alpha}\cdot\eta(u,w)^{\beta}}, & v\in C_u\\ 0, & \text{其他}\end{cases} \tag{7-10}$$

其中 c_k 表示在第 k 步选取的属性，C_u 表示未被选取属性所组成的集合。$\tau(u,w)$ 和 $\eta(u,w)$ 分别是边 (u,w) 上的信息素和启发信息。α 和 β 是两个参数，它们分别控制信息素和启发信息的相对重要程度。如果以下两个条件之一满足，构造过程就结束。

(1) 已选取属性的数目大于当前最小基数；

(2) $r_R(D)=r_C(D)$，其中 R 是蚂蚁构造的一个解。

第一个条件意味着不可能构造更好的解，因此构造过程没有必要继续。第二个条件意味着蚂蚁已经构造了一个更好的解。

在图 7-1 中显示了一个典型的构造过程，其中条件属性集 C 为 $\{a_1, a_2, a_3, a_4, a_5, a_6\}$，构造出的解是 $\{a_1, a_3, a_6, a_4\}$。在第一步，属性 a_1 被随机选取，随后属性 a_3 和 a_6 被选取，最后属性 a_4 被选取。在构造解的过程中，采用文献[14]中的方法来计算依赖度。

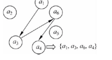


图 7-1　边模式蚁群算法构造过程示意图

当所有蚂蚁都完成构造过程时，信息素主要依据 MMAS 提供的更新规则来进行更新，并且考虑文献[11]中的建议。在边模式蚁群算法中(图 7-2 中为边模式的示意图，其中解是 $\{a_1, a_3, a_6, a_4\}$)，只有当前最优解 s_{gb} 对应路径上的信息素才被更新，即连接 s_{gb} 的前后两个属性的边上的信息素才被更新。边模式最初在经典文献[15]中应用。在这种模式下，信息素按照以下方法更新：

$$\tau(a,b)^{l+1}=\rho\tau(a,b)^{l}+\Delta\tau(a,b) \tag{7-11}$$

$$\text{如果 } \tau(a,b)^{l+1}>\tau_{\max},\quad \text{则 } \tau(a,b)^{l+1}=\tau_{\max} \tag{7-12}$$

$$\text{如果 } \tau(a,b)^{l+1}<\tau_{\min},\quad \text{则 } \tau(a,b)^{l+1}=\tau_{\min} \tag{7-13}$$

$$\Delta\tau(a,b)=\begin{cases} q/L_{gb}, & (a,b)\text{ 属于最优蚂蚁经过的路径} \\ 0, & \text{其他} \end{cases} \tag{7-14}$$

其中 $a,b\in C$，$a\neq b$，且 a,b 先后被选取。q 是一参数，$L_{gb}=|s_{gb}|$，$\tau_{\max}$ 和 $\tau_{\min}$ 分别是信息素上下界，$\tau(a,b)^l$ 是边 (a,b) 在第 l 代时的信息素。注意到文献[11]考虑解 s 的适合度(goodness)，即 $r_s(D)$。但由于 s_{gb} 的适合度 $r_{s_{gb}}(D)$ 是一个常数，因此解的适合度可以不考虑。由边模式的信息素更新规则可知，其时间复杂度和空间复杂度都为 $O(|C|^2)$。

7.3.2　团模式蚁群算法

在团模式中，每一个边都赋有一定的信息素和启发信息，且信息素和启

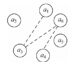


图 7-2　边模式信息素释放方式示意图

发信息的定义与边模式中一致，但是，蚂蚁信息素释放到连接其构造解中任意两个属性的边上。图 7-3 中给出了团模式示意图，其中解是$\{a_1, a_3, a_6, a_4\}$。团模式的基本思想是：虽然解的构造过程是有序的，但是其相应属性的任意排列都是可行解。在构造解时，蚂蚁从一个随机选取的属性出发，它也依据式(7-10)给出的概率选取下一个属性。



图 7-3　团模式信息素释放方式示意图

注意文献[16]中考虑用下式给出的概率选取下一个属性：

$$p(c_{k+1}=v \mid \tau, S_k)=\begin{cases}\dfrac{\left(\sum\limits_{u\in S_k}\tau(u,v)\right)^{\alpha}\cdot\left(\sum\limits_{u\in S_k}\eta(u,v)\right)^{\beta}}{\sum\limits_{w\notin S_k}\left[\left(\sum\limits_{u\in S_k}\tau(u,w)\right)^{\alpha}\cdot\left(\sum\limits_{u\in S_k}\eta(u,w)\right)^{\beta}\right]}, & v\notin S_k\\ 0, & \text{其他}\end{cases} \tag{7-15}$$

其中 c_{k+1} 表示在第 $k+1$ 步选取的属性，S_k 表示已被选取属性组成的集合。$\tau(u,w)$和 $\eta(u,w)$分别是边(u,w)上的信息素和启发信息。由式(7-15)可知，每个未被选属性在第 $k+1$ 步被选中的概率依赖于连接它与所有已被选中属性之间的边上的信息素和启发信息。前期的实验表明，采用这个概率来选取属性并没有明显改善计算结果，却增加了计算时间。因此本章仅用式(7-10)给出的概率来选取属性。构造过程在条件(1)或(2)两者之一满足时结束。

当所有蚂蚁都完成构造过程时，只有当前最优蚂蚁才能更新信息素。

在团模式下,所有连接 s_{gb} 每两个属性的边上的信息素被更新。具体而言,信息素按照如下方式更新:

$$\tau(a,b)^{t+1} = \rho\tau(a,b)^{t} + \Delta\tau(a,b) \tag{7-16}$$

$$\text{如果 } \tau(a,b)^{t+1} > \tau_{\max}, \quad \text{则 } \tau(a,b)^{t+1} = \tau_{\max} \tag{7-17}$$

$$\text{如果 } \tau(a,b)^{t+1} < \tau_{\min}, \quad \text{则 } \tau(a,b)^{t+1} = \tau_{\min} \tag{7-18}$$

$$\Delta\tau(a,b) = \begin{cases} q/L_{gb}, & a,b \in s_{gb}, \text{且 } a \neq b \\ 0, & \text{其他} \end{cases} \tag{7-19}$$

其中 $a,b \in C$, $a \neq b$, q 是一个参数, $L_{gb} = |s_{gb}|$, $\tau(a,b)^{t}$ 是边 (a,b) 在第 l 代时的信息素。与边模式一样,团模式的信息素更新规则的时间复杂度和空间复杂度都为 $O(|C|^{2})$。

7.3.3 点模式蚁群算法

在点模式中,每一个点都赋有一定的信息素和启发信息,并且仍然利用基于依赖度的启发信息。同样地,如果某个点的启发信息过小,则它被选取的概率就可能过小。为避免这种情形,有必要设置启发信息的取值范围。$\forall a \in C$, 启发信息 $\eta(a)$ 定义如下:

$$\eta(a) = |POS_{\{a\}}(D)| / |U| \tag{7-20}$$

$$\text{如果 } \eta(a) < \varepsilon, \quad \text{则 } \eta(a) = \varepsilon \tag{7-21}$$

其中 ε $(0<\varepsilon<1)$ 是一个正参数。

在构造解时,蚂蚁从一个随机选取的属性出发,它依据式(7-22)给出的概率选取下一个属性:

$$p(c_{k+1} = v \mid \tau) = \begin{cases} \dfrac{\tau(v)^{\alpha} \cdot \eta(v)^{\beta}}{\sum\limits_{w \in C_u} \tau(w)^{\alpha} \cdot \eta(w)^{\beta}}, & v \in C_u \\ 0, & \text{其他} \end{cases} \tag{7-22}$$

其中 c_k 表示在第 k 步选取的属性, C_u 表示未被选取属性组成的集合。$\tau(w)$ 和 $\eta(w)$ 分别是点 w 上的信息素和启发信息。由(7-22)可知,那些具有较高启发信息和信息素的属性被选中的概率较大。构造过程在条件(1)或(2)两者之一满足时结束。

在点模式蚁群算法中,蚂蚁将释放信息素到当前最优解对应的点上。信息素按照如下方法更新[19]:

$$\tau(a)^{t+1} = \rho\tau(a)^{t} + \Delta\tau(a) \tag{7-23}$$

$$\text{如果 } \tau(a)^{t+1} > \tau_{\max}, \quad \text{则 } \tau(a)^{t+1} = \tau_{\max} \tag{7-24}$$

$$\text{如果 } \tau(a)^{t+1} < \tau_{\min}, \quad \text{则 } \tau(a)^{t+1} = \tau_{\min} \tag{7-25}$$

$$\Delta\tau(a)=\begin{cases}q/L_{gb}, & a\in s_{gb}\\ 0, & \text{其他}\end{cases} \tag{7-26}$$

其中 $a\in C$，q 是一个参数，$L_{gb}=|s_{gb}|$，$\tau(a)^{l}$ 是属性 a 在第 l 代时的信息素。由点模式的信息素更新规则可知，其时间复杂度和空间复杂度都为 $O(|C|)$。

7.4 实验分析

为了测试算法的性能，本节将 ACO 应用到不同算例，它们的属性数和对象数不同。3 种算法都在 3GHz CPU(1GB RAM)PC 上运行。按照文献[11]中的建议，$\alpha=1$，$\beta=0.1$，$\rho=0.9$，信息素初始值为 0.5 且增加一个小的随机扰动，$\tau_{\max}=1$，$\tau_{\min}=0.001$，$q=0.1$。其他参数为 $\varepsilon=0.01$，蚂蚁数为 10 且最大运行代数为 100。

考虑文献[11]中的 13 个测试算例，其中 M-of-n、Exactly 和 Exactly2 来自于文献[17]，其他来自于 UCI data (Blake and Merz，1998)[18]。每个算例测试 20 次。

首先比较 4 种基于蚁群算法的属性约简算法：AntRSAR[11] 和 Edge-ACO、Clique-ACO、Vertex-ACO。注意到在 AntRSAR 中，蚂蚁数为 $|C|$，运行代数为 250。计算结果在表 7-2 中给出。对于本章 3 个算法，表中给出了平均基数以及时间(s)，而文献[11]中没有明确给出 AntRSAR 的计算时间。粗体字给出的是最好结果。

表 7-2 Edge-ACO、Clique-ACO 和 Vertex-ACO 实验结果

数据集	AntRSAR	Edge-ACO		Clique-ACO		Vertex-ACO	
	平均基数	平均基数	时间(s)	平均基数	时间(s)	平均基数	时间(s)
M-of-n	**6.00**	**6.00**	0.42	**6.00**	0.41	**6.00**	**0.32**
Exactly	**6.00**	**6.00**	0.46	**6.00**	0.44	**6.00**	**0.36**
Exactly2	**10.00**	**10.00**	0.97	**10.00**	0.96	**10.00**	**0.87**
Heart	6.10	**6.00**	0.34	**6.00**	0.34	6.05	0.34
Vote	**8.00**	**8.00**	0.28	**8.00**	0.29	**8.00**	**0.28**
Credit	8.60	8.60	1.36	**8.15**	1.25	8.55	**1.17**
Mushroom	**4.00**	**4.00**	1.56	**4.00**	1.46	**4.00**	**1.41**
Led	5.25	5.10	0.88	**5.00**	0.95	**5.00**	**0.75**
Letters	**8.00**	8.15	0.27	**8.00**	0.27	**8.00**	0.27
Derm	6.15	6.50	0.69	**6.00**	**0.64**	6.10	**0.64**
Derm2	8.85	9.40	0.73	**8.80**	0.68	8.85	**0.67**
Wq	13.45	14.55	1.38	**13.25**	1.32	13.40	**1.23**
Lung	**4.00**	**4.00**	0.32	**4.00**	0.32	**4.00**	**0.30**

由实验结果可见，Clique-ACO 的计算结果最好，Vertex-ACO 稍逊于 Clique-ACO，而 Edge-ACO 在 3 个算例中获得的结果比 AntRSAR 更好。由于算法的运行时间主要由解的依赖度计算时间决定，因此，可以把解的构造次数作为一个衡量算法计算时间的量。本章中算法构造解的总数为 1000，这小于 AntRSAR 构造解的总数。例如，数据集 M-of-n 的属性总数(其值为 13)最少，则其构造解的总数为 3250，而数据集 Lung 的属性总数(其值为 56)最多，则其构造解的总数为 14000。由于本章算法构造解的数目都小于 AntRSAR，因此 Clique-ACO 和 Vertex-ACO 能快速地找到最好解。

由表 7-2 中给出的计算时间可见，Vertex-ACO 的计算时间最小，这是因为在信息素更新时，Edge-ACO 和 Clique-ACO 的计算量大。另外，Clique-ACO 比 Edge-ACO 花费时间少，这是由于 Clique-ACO 能更快地收敛到较好解(相应地，其基数就小)，这样花费在计算依赖度的时间就少。由于 Vertex-ACO 和 Clique-ACO 性能较好，以后只考虑这两种算法。

下面与文献中其他元启发算法相比较，这些算法包括 GenRSAR[11](遗传算法)、SimRSAR[11](模拟退火算法)、TSAR[12](禁忌搜索)。这些算法的实验结果在表 7-3 中给出。对于每个算法，给出了实验所得的不同基数以及得到相应基数的次数(次数在括号中给出)。由结果可见，Vertex-ACO 和 Clique-ACO 在所有数据集合中的计算结果最好。GenRSAR 性能最差，TSAR 略优于 SimRSAR。关于生成解的个数，这些算法的粗略排名为：ACO＜TSAR＜SimRSAR＜GenRSAR。

表 7-3　Vextex-ACO、Clique-ACO、GenRSAR、SimRSAR 和 TSAR 的实验结果

数据集	Vextex-ACO	Clique-ACO	GenRSAR	SimRSAR	TSAR
M-of-n	**6**	**6**	$6^{(6)}\,7^{(12)}$	**6**	**6**
Exactly	**6**	**6**	$6^{(10)}\,7^{(10)}$	**6**	**6**
Exactly2	**10**	**10**	$10^{(9)}\,11^{(11)}$	**10**	**10**
Heart	$6^{(19)}\,7^{(1)}$	**6**	$6^{(18)}\,7^{(2)}$	$6^{(29)}\,7^{(1)}$	**6**
Vote	**8**	**8**	$8^{(2)}\,9^{(18)}$	$8^{(15)}\,9^{(15)}$	**8**
Credit	$8^{(12)}\,9^{(5)}\,10^{(3)}$	$\mathbf{8^{(17)}\,9^{(3)}}$	$10^{(6)}\,11^{(14)}$	$8^{(18)}\,9^{(1)}\,11^{(1)}$	$8^{(13)}\,9^{(5)}\,10^{(2)}$
Mushroom	**4**	**4**	$5^{(1)}\,6^{(5)}\,7^{(14)}$	**4**	$4^{(17)}\,5^{(3)}$
Led	**5**	**5**	$6^{(1)}\,7^{(3)}\,8^{(16)}$	**5**	**5**
Letters	**8**	**8**	$8^{(8)}\,9^{(12)}$	**8**	$8^{(17)}\,9^{(3)}$
Derm	$6^{(18)}\,7^{(2)}$	**6**	$10^{(6)}\,11^{(14)}$	$6^{(12)}\,7^{(8)}$	$6^{(14)}\,7^{(6)}$
Derm2	$8^{(3)}\,9^{(17)}$	$\mathbf{8^{(4)}\,9^{(16)}}$	$10^{(4)}\,11^{(16)}$	$8^{(3)}\,9^{(7)}$	$8^{(2)}\,8^{(14)}\,10^{(4)}$
Wq	$12^{(2)}\,13^{(8)}\,14^{(10)}$	$\mathbf{12^{(2)}\,13^{(11)}\,14^{(7)}}$	16	$13^{(16)}\,14^{(4)}$	$12^{(1)}\,13^{(13)}\,14^{(6)}$
Lung	**4**	**4**	$6^{(8)}\,7^{(12)}$	$4^{(7)}\,5^{(12)}\,6^{(1)}$	$4^{(6)}\,5^{(13)}\,6^{(1)}$

注：对于每个算法，给出了实验所得的不同基数以及得到相应基数的次数(次数在括号中给出)。

实验结果表明,采用点模式或者团模式的蚁群算法能高效地求解属性约简。从根本上而言,蚁群算法是构造性算法。在每一步,蚂蚁选取较好的属性,这样找到高质量解的可能性较大。而且,它用一个简单的机制来平衡蚂蚁探索和开发能力:它增加当前最优解对应的信息素,从而使蚂蚁能在这个最优解的邻域探索,同时,它通过限制信息素的上下界来避免早熟。另一方面,蚁群算法能快速构造解。这主要是由于采用了条件(1)或(2),尽可能少得计算依赖度。

7.5 小结

本章提出了3种求解属性约简的蚁群算法。它们分别采用边模式、团模式、点模式来释放信息素。实验结果表明,点模式和团模式是两个有效的信息素释放方式。点模式蚁群算法计算时间较少,但团模式蚁群算法计算结果较好。与其他元启发算法相比较,点模式蚁群算法和团模式蚁群算法能有效地找到基数较小的解。

也可以推广蚁群算法来解决其他种类的约简,例如近似熵约简[20]。另外,由于经典粗糙集理论不能很好地处理实值属性,有必要研究模糊粗糙集理论(fuzzy-rough set theory)[21]和蚁群算法来提取特征。至于大规模数据集,我们将进一步研究减少内存开销和计算时间的方法。

参考文献

[1] Theodoridis S, Koutroumbas K. Pattern Recognition[M]. Academic Press, 2006.

[2] Pawlak Z. Rough sets[J]. International Journal of Computer and Information Sciences, 1982, 11: 341～356.

[3] Pawlak Z. Rough Sets: theoretical aspects of reasoning about data[M]. Kluwer, Boston, 1991.

[4] Pawlak Z. Rough sets and data analysis[C]. Proceedings of the Asian fuzzy systems Symposium, 1996, pp. 1～6.

[5] Pawlak Z. Skowron A. Rudiments of rough sets[J]. Information Sciences, 2007, 177: 3～27.

[6] Skowron A, Pal SK. Rough Sets, Pattern Recognition, and Data Mining[J]. Pattern Recognition Letters, 2003, 24: 829～933.

[7] Swiniarski RW, Skowron A. Rough set methods in feature selection and recognition[J]. Pattern Recognition Letters, 2003, 24: 833～849.

[8] Wong SKM, Ziarko W. On optional decision rules in decision tables[J]. Bulletin of

Polish Academy of Science，1985，33：693～696.

[9] Wang X，Yang J，Teng X，et al. Feature selection based on rough sets and particle swarm optimization[J]. Pattern Recognition Letters，2007，28：459～471.

[10] Wróblewski J. Finding minimal reducts using genetic algorithms[C]. Proceedings of Second Annual Join Conference on Information Sciences，1995，pp. 186～189.

[11] Jensen R，Shen Q. Finding rough set reducts with Ant Colony Optimization[C]. Proceedings of 2003 UK Workshop Computational Intelligence，2003，pp. 15～22.

[12] Hedar A，Wang J，Fukushima M. Tabu search for attribute reduction in rough set theory[R]. Technical Report 2006-008，Department of Applied Mathematics and Physics，Kyoto University 2006.

[13] 王国胤. Rough 集理论与知识获取[M]. 西安：西安交通大学出版社，2001.

[14] Nguyen SH，Nguyen HS. Some efficient algorithms for rough set methods[C]. Proceedings of the Conference of Information Processing and Management of Uncertainty in Knowledge-Based Systems，1996，pp. 1451-1456.

[15] Dorigo M，Maniezzo V，Colorni A. Ant system：Optimization by a colony of cooperating agents[J]. IEEE Transactions on System Man，and Cybernetics-Part B，1996，26：29～41.

[16] Solnon C，Bridge D. An ant colony optimization meta-heuristic for Subset selection problems [C]. In System Engineering using Particle Swarm Optimization，Nedjah N，Mourelle L. Eds.，NY：Nova Science publisher，2006，pp. 7～29.

[17] Raman B. Ioerger TR. Instance-based filter for feature selection[J]. Journal of Machine Learning Research，2002，1：1～23.

[18] Blake CL，Merz CJ. 1998. UCI Repository of Machine Learning Databases[OL]，University of California at Irvine，http://www. ics. uci. edu/～mlearn/.

[19] Stützle T，Hoos HH. MAX-MIN ant system[J]. Future Generation Computer Systems，2000，16(8)：889～914.

[20] Slezak D，Wroblewski J. Order based genetic algorithms for the search of approximate entropy reducts[J]. Lecture Notes in Artificial Intelligence，2003，2639：308～311.

[21] Jensen R，Shen Q. Semantics-preserving dimensionality reduction：rough and fuzzy-rough-based approaches[J]. IEEE Transactions on Knowledge and Data Engineering，2004，16：1457～1471.



卫星资源调度问题

8.1 问题描述

卫星在现代生活中的应用越来越广泛，而卫星的正常运行离不开地面工作人员的维护。卫星测控是工作人员借助地面通信设备，向卫星发送控制指令和接收卫星数据的过程，它是卫星正常运行的重要保障[1,2]。

8.1.1 卫星测控基本概念

卫星在长期运行的过程中可能会出现偏离运行轨道或姿态发生变化等情况，所以需要地面工作人员定期对其轨道、姿态、星载实验设备状态等信息进行测量，若发现问题，还要借助通信设备对其发送控制指令，帮助其恢复正常状态；此外，卫星所收集到的数据也要定期传回地面。这些工作都是通过卫星测控完成的。

地面同卫星通信都是借助测控设备以无线电波的形式完成的。地面测控设备要同卫星进行通信，必须满足以下 3 个条件。

1）测控设备同卫星可见

前面已经提到，地面测控设备与卫星通信是借助无线电波实现的，由于无线电波只能够沿直线传播，因此只有当测控设备同卫星直线可见时，二者才能够建立通信链路进行数据交换。一般而言，固定在地面某处的测控设

备，只能在卫星绕地球运转一个周期中的部分时间段与其可见（同步卫星除外），这一可见时间段称为可见弧段或可见时间窗口（见图 8-1）。由此可知，每一个可见时间窗口都唯一对应一颗卫星和一个测控设备。

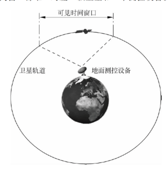


图 8-1　测控设备与卫星性可见示意图

在卫星绕地球运行一个周期内，一个地面固定的测控站与该卫星最多只能有一个可见时间窗口。至于地球同步卫星，由于其与地面相对静止，故对其可见的测控设备将始终可见。

2）测控设备同卫星的技术参数匹配

地面测控设备在与可见卫星进行通信时，首先要建立通信链路，即在测控设备天线与卫星天线之间建立无线通信链路，而无线通信链路的建立则要求测控设备和卫星的相关技术参数是匹配的。对测控设备和卫星之间通信链路建立有影响的技术参数主要有两类：测控设备和卫星所用的无线通信技术是否一致；卫星所在轨道是否在测控设备覆盖范围内。

由于无线通信技术在不断发展，在不同时期发射的卫星，其所使用的无线通信技术或相关参数，如所采用的无线电波的频段，是有差异的。同样，不同时期建造的地面测控站，其测控设备所使用的无线通信技术或相关参数也是有差异的。所以，在对卫星进行测控时，必须选用使用相同无线通信技术及相关参数的测控设备，即要满足测控设备和卫星所用的无线通信技术的一致性。

此外，不同类型的卫星，由于其所要完成的任务的要求不同，其运行轨道的高度有所不同。按照卫星的运行轨道高度的差异，可将卫星分为低轨

卫星、中轨卫星和高轨卫星 3 类（表 8-1）。不同轨道高度的卫星，对测控设备的要求也有所不同。一般而言，运行轨道越高的卫星，其测控难度越大，对测控设备的要求越高，反映在测控天线上，就是所需测控天线的口径越大。因此，在对不同轨道高度的卫星进行测控时，除了卫星对测控设备可见之外，还要保证卫星运行轨道在所选测控设备覆盖范围内。

表 8-1　低轨、中轨、高轨卫星的划分

卫 星 类 型	轨道高度(km)	运行周期(h)	典 型 代 表
低轨卫星	200～1200	2～4	侦察卫星
中轨卫星	20 000	12	导航卫星
高轨卫星	35 786	24	通信卫星

3）测控设备同卫星均处于空闲状态

在现有的测控系统中，一般而言，一个测控天线同一时刻只能同一颗卫星进行数据通信，卫星也是如此，即测控过程对测控设备和卫星都具有独占性。因此，一对相互可见且技术参数匹配的测控设备和卫星，只有当彼此都没有在执行其他测控任务时，才能够建立通信链路进行数据通信。

对卫星进行一次测控的具体过程如图 8-2 所示。首先，根据测控需求为卫星分配合适的测控设备和可见时间窗口；随后，测控设备在进行测控

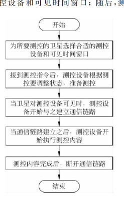


图 8-2　卫星测控过程流程

之前,根据测控任务的要求,调整测控设备状态,如测控天线的指向;当卫星相对测控设备可见时,测控设备开始尝试同卫星建立通信链路;通信链路建立之后,测控设备开始执行测控内容,一般包括发送控制指令、接收卫星传输的收据等;最后,当测控内容完成后,测控设备断开与卫星间的通信链路,重新回到空闲状态,至此整个测控过程结束。

8.1.2 卫星测控资源调度

通过前面的介绍可知,对于卫星正常运行,卫星测控是不可或缺的。随着在轨运行的卫星数量的不断增加,卫星测控需求也不断增加,这就为由分布在全球各地的测控设备组成的测控网络带来了挑战。要解决不断增加的测控需求同有限的测控资源之间的矛盾,有两种可行的方法:第一,增加测控设备的数量;第二,合理安排测控任务,提高测控设备使用效率。由于新建地面测控站增添测控设备的经济代价较大,如何对现有测控资源进行合理调度,提高其利用效率便受到了诸多关注。

在卫星日常运行维护过程中,工作人员会根据卫星的测控需求,生成一系列测控任务,每一个测控任务便是对某一颗卫星的一次测控过程。卫星测控资源调度即是借助优化算法,针对不同卫星的测控任务,分别选择合适的测控设备,最大限度地满足各卫星的测控需求。前面已经提到,只有当卫星对测控设备可见时,测控设备才能够对其进行测控。因此,测控任务必须在可见时间窗口内完成,或者占用整个可见时间窗口,或者只占用某个可见时间窗口的一部分。在本章中,将完整地执行一次测控过程所占用的时间段称为任务时间窗口(见图 8-3)。测控资源调度就是要为不同卫星的测控任务分配任务时间窗口,但是,在调度的过程中要注意不同可见时间窗口间

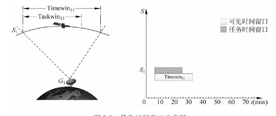


图 8-3 任务时间窗口示意图

的冲突情况。

所谓可见时间窗口冲突，是指多个可见时间窗口在时间上相互重叠的情况。由于测控任务对卫星和测控设备都有独占性，因此，在可见时间窗口的冲突部分最多只能够分配一项测控任务。可见时间窗口冲突可分为两类：第一类是多个卫星同时对某个测控设备可见造成可见时间窗口出现重叠(见图 8-4)；第二类是某颗卫星同时对多个测控设备可见造成可见时间窗口重叠(见图 8-5)。其中，第一类冲突情况是由测控过程对测控设备的独占性造成的，而第二类冲突情况是由测控过程对卫星的独占性造成的。

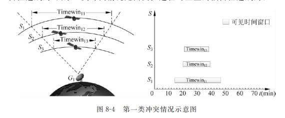


图 8-4　第一类冲突情况示意图

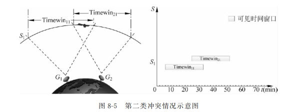


图 8-5　第二类冲突情况示意图

通过上面的介绍可以看到，简单地说，卫星测控资源调度就是针对不同卫星的测控任务，在可能存在冲突的可见时间窗口中，选择合理的、互不冲突的时间窗口。

8.2　卫星测控资源调度模型

在现有卫星测控资源调度相关问题的研究成果中，已经形成了多种用以描述卫星测控资源调度问题的数学模型。在本小节中，将介绍本章所用

卫星测控资源调度问题模型的建立过程及最终数学模型。

在介绍卫星测控资源调度模型之前，首先介绍模型中涉及的参数及变量。所用变量及其符号表示如下：

S 是所有需要测控的卫星集合，$S=\{1,2,\cdots,S\}$，$s\in S$ 代表某颗卫星；

M 是所有可用测控设备的集合，$M=\{1,2,\cdots,M\}$，$m\in M$ 代表某个测控设备；

J 是所有测控任务的集合，$J=\{1,2,\cdots J\}$，$j\in J$ 代表某个测控任务；

$s(j)$ 是任务 j 对应的卫星，因为每个测控任务都唯一对应一颗卫星，故 $s(j)\in S$；

$m(j)$ 是任务 j 可用的测控设备，因为一个测控任务可用的测控设备可能多于一个，所以 $m(j)$ 是一个集合，$m(j)\subseteq M$；

p_j 是执行测控任务 j 所需的时间，也称作任务持续时间；

w_j 是成功执行测控任务 j 所获得的收益；

T 是调度周期，即测控需求的时间跨度，其常见取值有 1 天或 1 周；

$V(s,m)$ 是在调度周期内，卫星 s 与测控设备 m 间所有可见时间窗口的集合

$$V(s,m)=\bigcup_{h=1,\cdots,H_{sm}}\left[t_{sm}^{start(h)},t_{sm}^{end(h)}\right] \tag{8-1}$$

其中，H_{sm} 为可见时间窗口的数量。

8.2.1 决策变量的选择

每一个可见时间窗口都有与之对应的卫星和测控设备，由于任务时间窗口肯定包含在某个可见时间窗口内，故对每一个测控任务而言，当确定了其相应的任务时间窗口之后，执行此测控任务的测控设备和时间便都已确定。任务时间窗口可以由任务开始时刻和任务持续时间唯一确定。所以确定每个测控任务的任务开始时刻，便能够唯一确定一个调度计划。

为了便于描述任务开始时刻，以分钟为单位将调度周期 T 划分成一组从 1 到 T 的时间段，每一段都可表示为 $[t-1,t)$，$t\in Z^{+}$。对于测控任务 j 而言，任务开始时刻的可能取值为

$$F'(j)=\bigcup_{m\in M(j)}Q(j,m) \tag{8-2}$$

其中，$Q(j,m)$ 表示测控设备 m 可以开始执行测控任务 j 的所有时刻的集合，$Q(j,m)\subseteq Z^{+}$。

$$Q(j,m)=\bigcup_{h=1,\cdots,H_{jm}}\left[t_{jm}^{start(h)},t_{jm}^{end(h)}-p_j\right] \tag{8-3}$$

定义二值变量 x_{jm}^{t}，$x_{jm}^{t}=1$ 表示测控设备 m 在 t 时刻开始执行测控任务 j，

其中 $t \in F'(j)$。卫星测控资源调度问题的优化过程便转化为，为每个测控任务 j 寻找合适的 m 和 t 使得 $x_{jm}^{t}=1$。

8.2.2　约束条件的描述

测控设备要对卫星进行测控，需要满足 3 个条件：测控设备同卫星可见、测控设备同卫星的技术参数匹配和测控设备同卫星均处于空闲状态。其中，在将卫星的测控需求转化为测控任务时，已经给出了每个测控任务的所有可用时间窗口和测控资源。此时，测控设备对卫星进行测控的前两个条件已经满足，在建立的数学模型中只需要保证，每个测控设备和每颗卫星在每一时刻最多只能执行一个测控任务。

首先，对测控设备而言，要保证测控设备在每一时刻最多执行一个测控任务，只需保证在每一个测控任务对应的任务时间窗口中，都不会有其他测控任务开始执行，所以，式(8-4)就保证了测控任务对测控设备的独占性。

$$\sum_{j=1}^{J}\sum_{\tau \in \Theta(j,m,t)} x_{jm}^{\tau} \leqslant 1, \quad m \in M, t \in T \tag{8-4}$$

其中，

$$\Theta(j,m,t) = [t, t+p_j-1] \cap F(j,m) \tag{8-5}$$

$$F(j,m) = \bigcup_{m \in M(j)} V(s(j),m) \tag{8-6}$$

$F(j,m)$ 表示任务 j 所有可用的可见时间窗口。

同样，对于卫星而言，要满足测控任务对卫星的独占性，只须保证针对同一颗卫星的测控任务的任务时间窗口不会出现重叠，即满足式(8-7)。

$$\sum_{m=1}^{M}\sum_{j \in J(s)}\sum_{\tau \in \Theta(j,m,t)} x_{jm}^{\tau} \leqslant 1, \quad s \in S, t \in T \tag{8-7}$$

8.2.3　卫星测控资源调度数学模型

卫星测控资源调度问题常用的优化目标有：

(1) 最大化测控任务调度成功率，即成功调度的测控任务数同总任务数之比；

(2) 最大化总收益，根据测控任务的重要程度赋予其不同的收益，最大化总收益值；

(3) 最大化测控设备利用率，尽可能地高效利用现有测控设备。

用 $f(x_{jm}^{t})$ 表示目标函数，结合前面的介绍，可以得到卫星测控资源调度问题的数学模型如下：

$$\max f(x_{jm}^{t}) \tag{8-8}$$

要求：

$$\sum_{m=1}^{M}\sum_{t\in F(j,m)} x_{jm}^{t} \leqslant 1,\quad j \in J \tag{8-9}$$

$$\sum_{j=1}^{J}\sum_{\tau\in \Theta(j,m,t)} x_{jm}^{\tau} \leqslant 1,\quad m \in M, t \in T \tag{8-10}$$

$$\sum_{m=1}^{M}\sum_{j\in J(s)}\sum_{\tau\in \Theta(j,m,t)} x_{jm}^{\tau} \leqslant 1,\quad s \in S, t \in T \tag{8-11}$$

$$\sum_{m=1}^{M}\sum_{t\in Q(j,m)} x_{jm}^{t} = \sum_{m=1}^{M}\sum_{t\in F(j,m)} x_{jm}^{t} \leqslant 1,\quad j \in J \tag{8-12}$$

$$x_{jm}^{t} \in \{0,1\} \tag{8-13}$$

其中，$Q(j,m)$ 的取值同式(8-3)，$\Theta(j,m,t)$ 的取值同式(8-5)。式(8-9)保证了每个测控任务最多只能被执行一次；式(8-10)和式(8-11)分别保证了测控任务对测控设备和卫星的独占性；式(8-12)保证了每个测控任务都只在一个可见时间窗口内完成。

本章将卫星测控资源调度问题描述成单目标优化问题。以测控任务调度成功率作为优化目标，即

$$f(x_{jm}^{t}) = \frac{1}{J}\sum_{j=1}^{J}\sum_{m=1}^{M}\sum_{t\in F(j,m)} x_{jm}^{t} \tag{8-14}$$

所以，卫星测控资源调度优化模型如下：

$$\max f(x_{jm}^{t}) = \frac{1}{J}\sum_{j=1}^{J}\sum_{m=1}^{M}\sum_{t\in F(j,m)} x_{jm}^{t} \tag{8-15}$$

要求条件满足式(8-9)～式(8-13)。

8.3 卫星测控资源调度问题求解

8.3.1 蚁群算法

蚁群优化算法[3]求解流程主要有两大步骤：路径构建和信息素更新，在解构建完成后可以附加局部搜索策略以改进所求得解的质量，求解框架如表 8-2 所示。

表 8-2　蚁群算法求解框架

Algorithm The Ant Colony Optimization
参数设置、初始化信息素初值
While 未达到算法结束条件　do
Ant_Solution_Construction
Apply_Local_Search(Optional)
Update_Phenomones
End While

算法初始化过程如表 8-3 所示，每只蚂蚁按表 8-4 所示过程独立完成解的构建后执行局部搜索算法对获得解的邻域进行搜索以提高解的质量，局部搜索过程如表 8-5 所示。

表 8-3　蚁群算法初始化过程伪码

Procedure Ant_Initialize
令迭代计数器 $t=0$
对所有弧(i,j)设置信息素初值 $\tau(i,j)=\tau_0$
For 每只蚂蚁 Do
根据具体问题，随机选择一个结点 c_1 作为起始结点
End For
其中，τ_0 为信息素初值，有 $\tau_{\min} \leqslant \tau_0 < \tau_{\max}$

表 8-4　蚁群算法解构建过程伪码

Procedure Ant_Solution_Construction

　　While $(s_k \in S \wedge s_k \notin S^*)$ 且 $(J(i) \neq \phi)$　do

　　蚂蚁在第 k 次迭代行走 i 步生成序列 $s_{ki} = <c_1, c_2, \ldots, c_i>$ 后，根据下式选取下一个结点 c_j。

$$P_{ij}(k) = \begin{cases} \dfrac{[\tau_{ij}(k)]^\alpha \cdot [\eta_{ij}(k)]^\beta}{\sum_{u \in J(i)} [\tau_{iu}(k)]^\alpha [\eta_{iu}(k)]^\beta}, & j \in J(i) \\ 0, & \text{其他} \end{cases}$$

式中：S^*——非空的最优解集合；

　　$J(i)$——从结点 i 可以直接到达的，又不在蚂蚁已经访问过的结点列表(禁忌列表)的结点集合；

　　$\tau_{ij}(k)$——第 k 次迭代时，弧(i,j)上的信息素值；

　　$\eta_{ij}(k)$——第 k 次迭代时，弧(i,j)上的启发式信息，启发式信息通常根据问题邻域信息进行定义；

　　α、β——$0<\alpha, \beta<+\infty$分别表示信息素和启发式信息的重要性参数。

　　在解的构建过程中，如果 $J(i)=\phi$，则蚂蚁完成了图的遍历，即完成了解的构建。

　　End While

表 8-5　蚁群算法局部搜索过程伪码

Procedure Apply_Local_Search
根据具体问题设计邻域结构对本次迭代局部优解 s_k 分别进行扰动
生成邻域解集合 $LS_k=\{ls_k^1, ls_k^2, \cdots, ls_k^{LS_k}\}$；
执行局部搜索启发式算法；
更新本次迭代局部最优解。

当所有蚂蚁构建完路径后，算法将会进行信息素更新。信息素的更新有两个步骤。首先，每次迭代后，蚂蚁都会根据自己构建的路径长度在它们本轮经过的路径上进行信息素局部更新。问题空间中所有路径上的信息素都会蒸发，信息素蒸发是自然界本身固有的特征，在算法中有助于避免信息素的无限积累，从而使搜索陷入局部最优。其次，当所有蚂蚁完成遍历后，则需进行信息素全局更新，信息素全局更新只在至今最优路径上蒸发和释放信息素。记 S_k^{ib} 为算法在第 k 次迭代中得到的最好可行解，S^* 为算法迭代至今得到的最好可行解。信息素更新过程如表 8-6 所示。

表 8-6　蚁群算法信息素更新过程伪码

Procedure Update_Phenomones

局部信息素更新：

当蚂蚁 k 完成图的遍历后，当且仅当其经过弧 (i,j) 时，对这条边按如下式进行信息素更新：

$$\tau_{ij}^k \leftarrow \min\{(1-\rho)\tau_{ij}^k + \Delta\tau_{ij}^k, \tau_{\max}\}$$

$$\Delta\tau_{ij}^k = \frac{Q}{L_k}$$

式中：

Q——信息素强度，它在一定程度上影响算法的收敛速度；

L_k——蚂蚁 k 在本次循环中所走的路径长度。

(2) 全局信息素更新：

$$\forall (i,j): \tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij}$$

如果 $f(s_k^{ib}) < f(s^*)$，那么 $s^* \leftarrow s_k^{ib}$

$$\forall (i,j) \in s^*: \tau_{ij} \leftarrow \min\{\tau_{\max}, \max\{\tau_{\min}, \tau_{ij} + \Delta\hat{\tau}\}\}$$

式中：

ρ——$0<\rho<1$ 为信息素蒸发率；

$\tau_{\min}$、$\tau_{\max}$——信息素下限与上限，f 为目标函数；

$\Delta\hat{\tau}=1/f(\hat{s})$；

s_k^{ib}——第 k 次迭代获得的最优解；

s^*——至今最优解。

8.3.2 解的构造

在实现过程中以每个任务的任务号所组成的序列的不同组合作为不同的解，在将一个序列转化为问题的解时，需要考虑3个要素：唯一性、解的质量和计算效率。

(1) 唯一性：在将一个给定的序列转化为问题的解时，一定要保证每一个序列都唯一对应一个问题的解，给定相同的序列，所得到的解也必须相同。只有保证了序列同解之间的唯一对应关系，才能够使进化过程中序列中所包含的解的优劣信息能够在迭代过程中得以传递，才能够使序列所对应的解的质量得到不断的优化。

(2) 解的质量：将一个序列转化为问题的解的方式有很多种，不同的方法所得到的解也不相同，而那些得到更高质量解的方法，能够在一定程度上加快整个算法的进化过程。所以，在将一个给定序列转化为问题的解时，要尽可能得到更高质量的解。

(3) 计算效率：由于将序列转化为问题的解的过程在算法中会被多次调用，所以，所采用方法的效率会对整个算法的效率产生很大的影响。从整个算法求解效率的角度来看，采用将序列转化为问题解的方法，需要具有较高的计算效率。在实际中，上面所说的3个要素通常不能同时得到满足。例如，要得到高质量的解，往往会增加计算量，使计算效率降低；而提高计算效率又常常会降低解的质量。所以，在选择将序列转化为问题的解的方法时，需要在解质量和计算效率之间做出选择。在蚁群算法中，选择了一种具有较高计算效率的方法：按照任务在序列中出现的顺序，依次为其分配测控资源。在为每一个任务分配测控资源时，并不进行回溯，而是直接将第一个可用的测控资源分配给它。当整个可用测控资源列表中都找不到满足某个测控任务要求的测控资源时，该任务将会被舍弃。

8.3.3 实验结果

实验采用西安卫星测控中心提供的实际测控数据(2015年3月30日到2015年4月5日)，共有83颗卫星，每天升降轨各两圈，总任务数量为2296。以下为实验结果：

设置挥发系数0.9，当蚂蚁数目为100，调度结果随迭代次数的变化如表8-7所示。

表 8-7 调度结果随迭代次数的变化

迭代次数	500	1000	1500	2000
结果	2289	2291	2291	2291

当迭代次数不变时(设置为 1000),调度结果随蚂蚁数目的变化如表 8-8 所示。

表 8-8 调度结果随蚂蚁数目的变化

蚂蚁数目	50	100	150	200
结果	2290	2290	2291	2291

可以看出,蚁群算法的效果总体较好。当然实际调度过程中的约束条件要复杂得多,可行解域会小很多,但蚁群算法仍然能在较短的时间内求出一个较优的可行解。

8.4 小结

卫星资源调度问题是在我国卫星资源调度实际需求基础上建立的一类调度问题。本章首先给出这类问题的数学模型,再利用蚁群算法来解决该问题。针对这类多约束问题,着重研究了算法中的解构造过程。

参考文献

[1] 康宁. 航天测控优化调度模型及其拉格朗日松弛求解算法[D]. 长沙:国防科学技术大学,2011.

[2] Wu Bin, Li Yuanxin, Huang Yong Xuan. Optimal Scheduling of TT&C Network Resources Based on Genetic Algorithm[J]. Journal of Astronautics, 2006, 27(6): 1136~1132.

[3] Stützle T, Hoos HH. MAX-MIN ant system[J]. Future Generation Computer Systems, 2000, 16(8): 889~914.

[4] 张娜. 多星测控资源调度问题建模及蚁群优化方法研究[D]. 西安: 西安交通大学, 2010.

[5] 杜挺. 基于拉格朗日松弛的卫星测控资源调度算法研究[D]. 西安: 西安交通大学,2013.



旅游路线规划问题

9.1 引言

随着人们生活水平的提高,越来越多的人选择外出旅游这一休闲方式,对旅游服务质量的要求也越来越高,个性化选择意识越来越强。一些研究者通过建立基于 WebGIS 的旅游推荐系统[1],试图快速地为旅行者提供景点的分布情况、道路的拥挤程度和景点的游览时间等信息,以便更便捷地服务旅行者,提升地区旅游服务质量,更合理地开发地区旅游资源。在这个系统中,一个关键问题是旅游路线规划问题。由于人们的生活节奏愈发加快,外出旅游也会有诸多限制,如时间、开销、景点偏好、路线等,导致该问题难以建模和优化。

9.2 问题描述

一般地,旅游路线规划问题可描述如下:在满足用户约束条件下(如往返交通、日程等安排,对费用、单日游览时间的需求,景点开放时间窗口、景点间交通耗时等),选择合适的路线,最大化用户满意度[2,3]。

在本章,假设每个景点都有一个给定收益值,用一个景点的收益值描述用户对该景点的偏好以及游玩该景点所得到的体验。旅游路线规划问题可

看作在有 V 个顶点和 E 条边的完全图 $G(V,E)$ 中寻找满足约束条件的最优解。其中，每个景点的信息都可表示为一个五元组 $\{S_i, s_i, C_i, c_i, T_i\}$，$S_i$ 表示收益值，s_i 与 c_i 分别表示景点的开放时间与关闭时间，C_i 是景点的入场门票费用，T_i 表示景点的建议游玩时间；每条边 t_{ij} 则表示两个景点之间的转移时间。最优路线需在满足时间、费用等因素的条件下，实现用户满意度值的最大化。

该问题与带有时间窗的团队定向问题具有一定的相似性，如景点开放时间窗口对应配送节点开放时间窗口，景点满意度对应配送节点的收益值，景点游览时间则对应节点的服务或访问时间。两者之间区别主要有以下方面。

(1) 约束条件存在差异。旅游路线规划问题除需要满足带有时间窗团队定向问题的约束条件外，还要满足用户预算所带来的约束条件，即门票费用不能超过用户输入预算信息。

(2) 路线体验不同。旅游作为享受旅途的过程，与实际团队定向问题完成最后一公里配送具有较大心理差别，主要体现在游客可能希望延长对当前景点的游览，往往对行程节奏有一定要求(如老年人往往希望节奏慢点)，而团队定向问题则必须严格根据时间约束进行。

(3) 优化目标不同。团队定向问题优化目标通常选取为收益最大化，而旅游路线规划问题中由于满意度可以不同的表现形式，如景点收益值之和最大化，成本最小化，或行程更轻松等。

9.3 旅游路线规划问题的数学模型

为了建模旅游路线规划问题，下面针对模型中相关因素进行分析。

1) 用户需求

用户需求主要包括用户对行程起始时间、结束时间、单日游览时间及出行费用的限制。其中单日游览时间表示每天可以用于游览的最大时间，如早上九点至晚上七点对应的单日游览时间则为 10 小时。单日游览时间上限记为 $t_{\max}$，费用开销上限记为 $C_{\max}$。

2) 景点描述信息

景点信息主要包括四个因素：景点开放时间窗、景点游览时间、景点费用、景点的收益值(注意收益值依赖于用户)。对于第 i 个景点，$[O_i, C_i]$ 表示景点开放时间窗口，即景点 i 的游览时间只能在这一时间区间内进行；景点游览时间表示景点推荐游览时长。采用时间区间来表征这一参数，即用

户在该景点的实际游览时间在一上下界之间，其中上下界分别记为 $T_{\max}$ 和 $T_{\min}$，实际游览时间记为 T_i；景点的门票费用记为 c_i；用户对景点的收益值则是用户对景点评分值，记为 S_i。

3）时间约束

时间因素除了景点开放时间窗口、游览时间外，还需要考虑景点之间转移时间 t_{ij} 及单日游览时间限制 $t_{\max}$ 等时间约束。

4）费用约束

此处费用仅考虑门票费用支出，所有门票支出需小于等于用户预算上限。

另外，引入紧凑度因子 δ 来描述用户对行程紧凑程度的需求，这一参数定义在区间[0,1]中，以0.5为分界点表征行程的紧凑程度。景点实际游览时间作为用户偏好值和紧凑度共同作用的函数，其函数表示为

$$T_i = F(T_{\max}, T_{\min}, \delta, S_i) \tag{9-1}$$

其中，$T_{\max}$ 和 $T_{\min}$ 分别表示景点推荐游览时间上界与下界。

当紧凑度较高时，将会有过多景点被规划至路线中，从而导致过多时间用于景点间往返交通。基于此，在目标函数中加入了紧凑度的评价指标。该指标选取为路线所包含的景点数和游览时间 T_{visit} 与总时间 T_{total} 的比率之和，在一定的行程紧凑度要求下尽可能保证足够多的游览时间。

$$\max \sum_{k=1}^{m}\sum_{i=2}^{n-1} S_i y_{ik} + \eta_1 \cdot (\delta - 0.5) \cdot \left(\sum_{k=1}^{m}\sum_{i=2}^{n-1} y_{ik} + \frac{T_{\text{visit}}}{T_{\text{total}}} \cdot \eta_2 \right) \tag{9-2}$$

$$T_{\text{visit}} = \sum_{k=1}^{m}\sum_{i=2}^{n-1} T_i y_{ik} \tag{9-3}$$

$$T_{\text{total}} = \sum_{k=1}^{m}\left(\sum_{i=1}^{n-1}\sum_{j>i} t_{ij} x_{ijk} + \sum_{i=1}^{n} t_i y_{ik} \right) \tag{9-4}$$

其中，n 表示所有景点的数量，η_1 和 η_2 分别表示行程内景点数及游览时间占总时间比重的加权值，用以控制紧凑度对解的影响，m 表示游客期望旅游的总天数，$\{S_i, s_i, C_i, c_i, T_i\}$ 表示景点相关的属性信息，y_{ik} 表示景点 i 是否被规划在第 k 天，x_{ijk} 表示线路 (i,j) 被规划在第 k 天的行程中时则为1，否则为0。

此外，最终路线结果还应满足时间、成本等约束条件，即最终成本应不超过用户所设定的 $C_{\max}$，每日游览时间不超过上限 $t_{\max}$，约束条件描述为

$$\sum_{k=1}^{m}\sum_{i=2}^{n-1} C_i y_{ik} \leqslant C_{\max} \tag{9-5}$$

$$\sum_{i=1}^{n-1}\sum_{j=2}^{n} t_{ij} x_{ijk} + \sum_{i=1}^{n} t_i y_{ik} \leqslant t_{\max} \quad k = 1,2,\cdots,m \tag{9-6}$$

$$s_{ik} + T_i + t_{ij} - s_{jk} \leqslant M^*(1 - x_{ijk}) \quad i,j = 1,2,\cdots,n; k = 1,2,\cdots,m \tag{9-7}$$

$$\sum_{i<j} x_{ijk} + \sum_{i>j} x_{ijk} = 2y_{jk} \quad (j = 2,3,\cdots,n-1; k = 1,2,\cdots,m) \tag{9-8}$$

$$\sum_{k=1}^{m} y_{ik} \leqslant 1 \quad (i = 2,3,\cdots n-1) \tag{9-9}$$

$$\sum_{j=2}^{n}\sum_{k=1}^{m} x_{1jk} = \sum_{i=1}^{n-1}\sum_{k=1}^{m} x_{ink} = m \tag{9-10}$$

$$O_i < s_{ik} < C_i \quad (1 \leqslant i < j \leqslant n; k = 1,2,\cdots,m) \tag{9-11}$$

$$x_{ijk} \in \{0,1\} \quad (1 \leqslant i < j \leqslant n; k = 1,2,\cdots,m) \tag{9-12}$$

$$y_{1k} = y_{nk} = 1 \quad y_{ik} \in \{0,1\} \quad (i = 2,3,\cdots,n-1; k = 1,2,\cdots,m) \tag{9-13}$$

在上述模型中,约束条件(9-5)表示用户的费用约束;式(9-6)表示每日总旅游时间约束;式(9-7)表示连续行程中后一个节点的游览应在前一个节点结束之后开始;式(9-8)~(9-10)表示除编号为 1 和 n 的景点作为出发地点与返回地点应被访问 m 次外,其他景点只能被游览一次;式(9-11)则表示景点的开始时间应满足景点开放时间窗口;式(9-12)与式(9-13)限定了变量的取值范围。

9.4 相关算法

9.4.1 GLS(Guided Local Search)

GLS 的基本思想是:当应用 GLS 求解问题时,需要针对候选解定义一系列的特征(如针对 TSP 问题,可选择解是否直接包含从 A 地至 B 地这一路线),在邻域搜索陷入局部最优时,将会选择符合的特征并惩罚,并加入到目标函数中,由此改变邻域搜索在搜索过程中的策略,跳出局部最优[4]。

给定目标函数 $g(s)$,邻域搜索过程中采用的带有惩罚项的目标函数 $h(s)$为

$$h(s) = g(s) + \lambda \times \sum_{i \in \text{features}} (p_i \times I_i(s)) \tag{9-14}$$

式中,λ 是惩罚系数,i 是特征,p_i 是第 i 个特征的惩罚因子,所有 p_i 最初均为 0,I_i 表示当前解 s 是否满足特征 i,满足则为 1,否则为 0。

对于最小化问题,具有较大代价的特征对解所造成的代价影响最大,当达到局部最优点时,将判断特征对当前局部最优解的代价所产生的作用。同时,为避免特征被多次惩罚,还将考虑惩罚因子的当前值。当前特征对解

所产生影响的计算方式为

$$util_i(s_*) = I_i(s_*) \times \frac{c_i}{1+p_i} \tag{9-15}$$

其中,c_i 是特征 i 的代价,p_i 表示当前惩罚值,即当前解中若不包含该特征,则惩罚项所产生的影响为0。同时,p_i 越大,其所产生的影响越小。

Souffriau 等[4]用 GLS 来求解一个旅游路线规划问题的简单模型,在50个节点以上的算例中得到的结果相比 DTGS(Dynamic Tour Guide systems)具有明显优势。其算法伪代码在表 9-1 中给出。

表 9-1 GLS 算法流程

```
Algorithm 3 GLS 算法流程
1: Construct;
2: Loop2=0;
3: while Loop2<MaxLoop2 do
4:    Loop2++;
5:    Loop1=0;
6:    while Loop1<MaxLoop1 and Solution improved do
7:       Loop1++;
8:       TSP;
9:       Insert;
10:      Replace;
11:   end
12:   if Solution better than BestFound then
13:      BestFound=Solution;
14:   else
15:      if Solution=BestFound then
16:         if Not Disturbed Before then
17:            Disturb;
18:         end
19:      end
20:   end
21:   if Loop2=MaxLoop2/2 then
22:      Disturb;
23:   end
24: end
25: Return BestFound;
```

在算法中，解构建算子 Construct 采用贪婪算法，它根据距离最短这一准则进行初始解的构建；TSP 算子是利用 GLS 求解 TSP（旅行商问题）时的启发式框架，通过迭代时调用 2-opt move 寻找序列中的最短路线，从而尽可能插入更多新的景点，获得更大的目标函数值；Insert 算子则是从未分配的景点中选择，并在满足游览时间限制的条件下，插入游览时间最短的景点；Replace 算子对当前解中具有较低评分的景点进行替换，当无法进行替换时，则当前解到达了局部最优。

为了跳出局部最优点，该算法应用了两个 GLS 惩罚项：①对未在当前解序列中的景点的评分增加一个激励项；②对解序列中的景点减少惩罚项。

所有未包含在路线中的结点对解的正面影响由式(9-16)计算，具有最大正面影响的结点将在分值中增加 p_R。

$$U_i = \frac{S_i'}{1 + nr_i} \tag{9-16}$$

其中，S_i'是当前结点分值与所得到的激励项之和：$S_i' = S_i + nr_i * p_R$；nr_i 表示结点 i 获得激励项的次数；p_R 表示激励项的大小；分母 $1+nr_i$ 的引入则是为避免所有结点均有机会得到激励项。

接受惩罚项的结点则是通过计算当前解中每个景点的负面影响来选择，并由式(9-17)计算得到

$$DU_i = \frac{1}{S_i' \cdot (1 + np_i)} \tag{9-17}$$

其中，$S_i' = S_i - np_i * p_R$，np_i 表示该结点被惩罚的次数。

当惩罚项与激励项被调用时，在下一轮迭代中将对解空间进行新一轮的搜索。

Disturb 算子则是通过随机移除每条行程中的部分景点，来保证所有结点均有机会被加入到解序列中。该算子会被运行至少一次，从而提高在解空间搜索过程中解的多样性。

9.4.2 GRASP(Greedy Random Adaptive Search Procedure)

GRASP 最早由 Feo 和 Resende 于 1989 年提出[5]，目前已被成功用于求解带有时间窗的团队定向问题[6]。Vansteenwegen[7]利用该算法实现了在线系统城市路线规划器，研究表明，该算法能以较高效率返回用户满意的路线结果。表 9-2 中介绍了该算法的主要流程。

表 9-2 GRASP 算法流程

Algorithm 4 GRASP 算法流程
1：**while** *Stopping criterium is not met* **do**
2： *Solution*=*empty*；
3： *Greediness*=*random*(0,1)；
4： *VisitList*=*GeneratePossibleVisits*(*Solution*)；
5： **while** *VisitList not empty* **do**
6：   **foreach** *Visit in VisitList* **do**
7：   Calculate heuristic value；
8：   **end**
9：   Determine threshold value；
10：   Restrict *VisitList*；
11：   *RandVisit*=*random Visit from restricted VisitList*；
12：   Insert RandVisit into *Solution*；
13：   *VisitList*=*GeneratePossibleVisits*(*Solution*)；
14： **end**
15：**end**
16：Return *BestFound*；

在满足终止条件前，该算法将反复迭代地对解空间进行搜索。每一个迭代过程都会随机选定一个贪婪系数，用于描述贪婪性与随机性之间的比率。随后，根据当前解中已包含的结点，确定可以加入到当前解的候选结点列表，并对候选列表中所有结点根据式(9-18)计算其启发信息

$$h_j = \left| \frac{\text{InterestScore}_j^2}{\text{Shift}_j} \right| \tag{9-18}$$

其中，Shift_j 表示从当前节点到节点 j 所需要的时间，InterestScore_j 则是当前景点的评分值。

同时，为提高计算效率，式(9-19)计算了过滤候选列表节点的阈值

$$\text{threshold} = (h_{\max} - h_{\min}) \cdot \text{greediness} \tag{9-19}$$

其中，$h_{\max}$ 和 $h_{\min}$ 分别表示当前候选列表所有结点的启发信息中的最大值与最小值，greediness 表示当前迭代周期的贪婪系数。

对于候选列表中的剩余结点，则随机选择一个结点加入当前解中。当序列中所有结点均无法在满足时间约束的条件下访问时，算法将会被终止。

9.4.3 烟花算法

烟花算法最早由 Ying Tan 提出并用来求解复杂函数的优化问题[8]。作为一种群体智能优化算法，烟花算法主要包括三个过程：解的初始化、邻

域搜索和解的选择机制。算法流程图如图 9-1 所示。

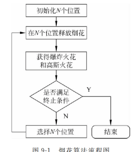


图 9-1 烟花算法流程图

1）解的初始化

在解空间里随机选择 N 个位置，作为初始烟花位置并启动烟花，每个烟花代表解空间中的一个可行解。

2）邻域搜索

在烟花启动后，将会在周围区域内产生一定数量的火花，这一过程可看作在邻域内进行采样过程。在这一阶段中，将会生成两种不同的火花，爆炸火花和高斯火花，前者主要对邻域空间进行搜索，后者则通过探索更大邻域空间，提高解的多样性。其具体细节描述如下。

（1）爆炸火花，烟花算法对算法的搜索能力和开采能力引入了平衡机制。以最小化目标函数模型为例，针对适应度值较好的烟花，即其目标函数值也更小，将选择更小的爆炸半径 A_i 和更大的爆炸火花数量 S_i，对于每个烟花 X_i，其爆炸半径与爆炸火花数量计算公式为

$$A_i = \hat{A} \cdot \frac{f(X_i - y_{\min}) + \varepsilon}{\sum_{i=1}^{N}(f(X_i) - y_{\min}) + \varepsilon} \tag{9-20}$$

$$S_i = M_e \cdot \frac{y_{\max} - f(X_i) + \varepsilon}{\sum_{i=1}^{N}(y_{\max} - f(X_i)) + \varepsilon} \tag{9-21}$$

其中，$y_{\min} = \min(f(X_i))$，$y_{\max} = \max(f(X_i))$，$\hat{A}$和 M_e 作为两个参数，分别用于调节搜索半径及爆炸火花数量，ε 作为极小量，用于避免除零操作。此外，为避免爆炸火花数量不合理，对其设定了上界和下界分别为 aM_e

和 bM_e。

(2) 高斯火花,在爆炸过程结束后,从烟花集合中随机选取 M_g 个烟花,并对其中随机选择 k 个维度与正态分布变量相乘,即

$$\hat{X}_{ik} = X_{ik} \cdot e \tag{9-22}$$

式中 $e \sim N(1,1)$,e 满足均值与方差均为 1 的正态分布。改进烟花算法[9](Enhanced Fireworks Algorithms)中对该算子加入了映射机制,在乘积超出边界后会随机将该维度映射在可行域内任意区域。

3) 选择机制

在邻域搜索结束后,新一代的烟花种群将会在候选集 K 中选出,候选集 K 由当前烟花种群、爆炸火花和高斯火花组成。候选集合中的最优解将会被直接保留至下一代烟花种群中。对于其他的候选解 X_i,则依概率 $P(X_i)$ 予以保留,$P(X_i)$ 表示为

$$P(X_i) = \frac{R(X_i)}{\sum_{X_j \in K} X_j} \tag{9-23}$$

$$R(X_i) = \sum_{X_j \in K} d(X_i - X_j) = \sum_{X_j \in K} || X_i - X_j || \tag{9-24}$$

式中,$R(X_i)$ 是当前候选解与所有其他候选解的距离之和。相比单纯依据目标函数值的轮盘赌选择策略,这一机制可以避免新一代种群过于集中在某一个或某几个局部最优点附近,从而保证了所筛选出的解具有一定的多样性,但却不利于保留候选集中除去当前最优解后的其他近优解。

考虑到连续问题与离散问题在距离和解空间上的差异,在用烟花算法求解 TTDP 时,要对距离及相关算子进行重新定义。记可行解序列为 $\pi = \{\pi_{11}, \pi_{12}, \cdots, \pi_{1k_1}, \pi_{21}, \cdots, \pi_{mk_m}\}$,其中 $\pi_{ip} \neq \pi_{jq}$,π_{dk} 表示第 d 天行程中的第 k 个景点。由于结点完全相同的两个解之间可以通过调整序列顺序相互转化,因此本文中对解之间的距离定义如下:

定义:解之间的距离定义为两个解序列不考虑顺序时所包含的不同结点的数目,即相同结点所组成的解序列之间的距离为 0。

在这一基础上,本文对经典烟花算法框架下算法的主要流程及所涉及算子进行了重新定义。

1) 解的初始化

烟花算法最初主要用于连续复杂函数的优化问题。对于这类问题,随机生成初始烟花,有利于保证解在解空间均匀分布。但对于旅游路线规划问题而言,随机生成一个序列,并将序列解析为可行解,有很大可能性会导致解的质量较差。因此采用随机性与贪婪性相结合的机制进行解的初

始化。

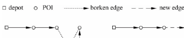
采用串行法进行构造解，首先对每个景点建立候选列表，候选列表的排序依据如下启发信息

$$\mathrm{Heur}_j = \frac{(S_i)^2}{t_{ij} * T_i} \quad i,j = 2,3,\cdots,n-1 \tag{9-25}$$

在构造解时，即构造 m 条单独的路线。对于每条路线，首先随机选择起点，并对该景点的候选列表按顺序进行判断，若尚未加入到行程中，则以一定概率 β(如 0.95)添加至该路线，并作为当前景点继续这一操作，直至该路线时间约束不能满足。

2）爆炸算子

爆炸算子的核心作用是在适应度值较好的解附近以更小的半径进行更多次数的邻域搜索，基于对于距离的定义，爆炸算子通过删除至多 A_i 个结点，再依次添加尽可能多的结点的方式，并利用 2-opt move 算子对解序列中单条路线进行调整，2-opt move 如图 9-2 所示。



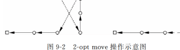


图 9-2　2-opt move 操作示意图

爆炸算子伪代码如表 9-3 所示。此外，由于路线规划模型为最大化问题，因此爆炸半径和爆炸火花数量将由式(9-26)与式(9-27)得到

$$A_i = \hat{A} \cdot \frac{y_{\max} - f(X_i) + \varepsilon}{\sum_{i=1}^{N}(y_{\max} - f(X_j)) + \varepsilon} \tag{9-26}$$

$$S_i = M_e \cdot \frac{f(X_i) - y_{\min} + \varepsilon}{\sum_{j=1}^{N}(f(X_j) - y_{\min}) + \varepsilon} \tag{9-27}$$

表 9-3　爆炸算子流程

Algorithm 1　爆炸算子流程
1: Set Parameters N, $\hat{A}$, M_e
2: Calculate the amplitude A_i and number of sparks S_i

续表

Algorithm 1 爆炸算子流程
3: **for** i = 1 to S_i **do**
4: $rand = Rand(A_i)$
5: **for** j = 1 to $rand$ **do**
6: Randomly remove a node from the current solution
7: Greedy insert new nodes at the best place
8: Iteratively choose a daily tour to operate the 2-opt move
9: **end for**
10: **end for**
11: Store all the explosion sparks into candidate set **K**

3）高斯算子

经典烟花算法中，高斯算子主要用于保证解的多样性。在求解连续问题时，通过对解的部分坐标进行变换，使其对更大的空间进行搜索。对于离散问题，这一过程难以实现。此节中提出的高斯算子致力于寻找当前解序列所包含景点的最佳顺序，同时允许插入新景点，以进一步提高目标函数值。高斯算子除采用 2-opt move 对单条路线进行优化外，还将采用 cross move 对不同路线进行优化。为保证解的多样性，该算子允许以一定概率 P_a 接受坏解。cross move 示意图如图 9-3 所示。

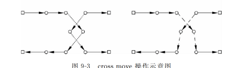


图 9-3 cross move 操作示意图

该算子伪代码描述如表 9-4 中所示。

表 9-4 高斯算子流程

Algorithm 2 高斯算子流程
1: Set Parameters N, M_g, P_a
2: **for** i = 1 to M_g **do**
3: Randomly select two nodes from two tours, operate cross move
4: $L = old\ profit$, $L' = new\ profit$;
5: **if** $(L < L')$

续表

Algorithm 2 高斯算子流程
6:　　Implement operation;
7:　**else**
8:　　$rand = Random(1)$;
9:　　**if** $(rand < P_a)$
10:　　　Implement operation
11:　　**endif**
12:　**endif**
13:　Iteratively choose a daily tour to operate the 2-opt move
14: **end for**
15: Store all the mutation sparks into candidate set **K**

4）选择机制

选择机制在烟花算法中扮演了重要角色。与经典烟花算法相似，同样直接保留当前烟花种群中的最优解至下一种群，对于其他解，则采用类似轮盘赌的方式进行选择。为增加解的适应度值在筛选中的权重，采用平方项，而不是一次项计算选择概率。每个解序列被选择的概率为

$$P_{\text{sel}}(X_i) = \frac{(f(X_i) - y_{\min})^2}{\sum_{X_j \in K} ((f(X_j) - y_{\min}) + \varepsilon)^2} \tag{9-28}$$

根据这一公式，具有更好适应度值的解，将更可能被保留至下一种群，相比增强型烟花算法中提到的随机选择机制，实验结果表明这一选择策略具有更好的求解效果。

9.5　蚁群算法及其分析

将第5章中求解定向问题的自适应最大最小蚁群算法(AMMAS)推广来解决TTDP。算法用串行法构造解，启发信息、信息素的管理方式与第5章相同。

为测试算法，构造四个算例，参数 η_1 和 η_2 均选择为10。通过统计分析，参数设置如下：在GLS中，内循环迭代次数设置为300，种群规模取为28。在烟花算法中构造解贪婪系数 β 取为0.95，爆炸火花半径上界为10，爆炸火花半径下界1，爆炸火花数量上界 aM_e 取为15，爆炸火花数量下界 bM_e 取值为5，高斯火花数量取值为10。在蚁群算法中，种群规模为20，其余参数与第5章相同。算法最大运行时间为3秒。

各算法运行所得到的最优目标函数值及二十次运行的平均目标函数值见表 9-5。由表中结果可见,AMMAS 是一种效果很好的算法,它在所有算例中都得到最好的最优值,而平均值在 3 个算例上好于其他算法。

表 9-5 四种算法结果对比

算例	FWA		GLS		GRASP		AMMAS	
	平均值	最优值	平均值	最优值	平均值	最优值	平均值	最优值
算例 1	366.3	367.1	366.6	369.1	366.2	368.2	367.1	369.1
算例 2	443.8	446.2	443.2	446.2	431.2	439.3	444.6	446.2
算例 3	392.8	393.2	392.2	393.2	390.2	391.2	392.7	393.2
算例 4	370.3	370.6	365.4	370.6	361.3	369.5	371.0	371.1

9.6 小结

本章探讨了旅游路线规划问题并给出了该问题的 0-1 整数规划模型。推广了求解定向问题的自适应蚁群算法,并与 GLS、GRASP 和烟花算法进行了对比。实验结果表明,AMMAS 算法是一种求解该问题的有效算法。

参考文献

[1] D Gavalas, C Konstantopoulos, K Mastakas, et al. A survey on algorithmic approaches for solving tourist trip design problems[J]. Journal of Heuristics, 2014, 20 (3): 291~328.
[2] Hui Ding, Liangjun Ke, Zihao Geng. Route Planning in a New Tourist Recommender System: a Fireworks Algorithm Based Approach [C]. 2016 IEEE Congress on Evolutionary Computation. Vancouver, Canada, 2016.
[3] 丁辉. 个性化旅游推荐系统中关键问题研究[D]. 西安交通大学,2015.
[4] W Souffriau, P Vansteenwegen, J Vertommen, et al. A Personalized Tourist Trip Design Algorithm for Mobile Tourist Guides[J]. Applied Artificial Intelligence, 2008, 22 (10): 964~985.
[5] Feo TA, Resende MG. A probabilistic heuristic for a computationally difficult set covering problem[J]. Operations research letters, 1989, 8 (2): 67~71.
[6] A Garcia, O Arbelaitz, MT Linaza, et al. Personalized tourist route generation [M]. German: Springer, 2010.
[7] P Vansteenwegen, W Souffriau, GV Berghe, et al. The City Trip Planner: An expert system for tourists[J]. Expert Systems with Applications, 2011, 38 (6):

6540～6546.

［8］ Y Tan，Y Zhu. Fireworks algorithm for optimization［J］. Advances in Swarm Intelligence. Springer，2010：355～364.

［9］ S. Zheng，A. Janecek，Y. Tan Enhanced fireworks algorithm［C］. IEEE，2013：2069～2077.



多目标组合优化问题

10.1 引言

在许多实际应用中，一个决策者往往要考虑多个相互冲突的目标，并且试图确定一个最优折中方案。这类应用可以建模成多目标优化问题。该问题的一个 Pareto 最优解就是一个最优折中的候选方案[1]。一个多目标优化问题可以有很多甚至是无穷多个 Pareto 最优解。在决策空间（或目标空间）所有的 Pareto 最优解构成 Pareto 前沿或解集。如果没有决策者的偏好信息，所有的 Pareto 最优解都是一样的。在实际应用中，决策者希望能足够好地逼近 Pareto 前沿以更好地分析问题，从而得到最终决策。

许多进化算法已经被用于求解多目标优化问题[2]。多目标进化算法的优势在于它们通过进化一组候选解可以在一次测试之后得到一个逼近 Pareto 最优解的解集。

蚁群算法是一类反应式搜索优化技术，它在优化的同时进行学习。它试图结合机器学习技术将搜索过程中在线信息融入智能计算方法并自主调节算法行为。蚁群算法最初是用于求解单目标组合优化问题。它利用信息素矩阵记录搜索过程中习得的知识。每个解元素的信息素值表征该解元素在一个好解中的可能性。为了引入问题相关信息，蚁群算法往往在搜索之前就定义一个启发信息矩阵。蚁群算法利用一组蚂蚁进行搜索。在每一

代,每个蚂蚁利用信息素和启发信息矩阵来构造解,然后利用新构造的解更新信息素矩阵的值。蚁群算法已经成功用以解决许多单目标优化问题,受此启发,文献[3～9]中已经有多目标蚁群算法的报道。在设计多目标蚁群算法时,两个相互关联的因素需要仔细考虑。

(1) 信息素和启发信息矩阵的定义:由于多目标蚁群算法的目标是很好地逼近整个 Pareto 前沿,信息素和启发信息矩阵应该包含所有 Pareto 最优解的可能位置而不是某个解的信息。如果仅仅采用一个信息素矩阵和一个启发信息矩阵,难以服务于这一目标,尤其是在 Pareto 前沿的分布非常复杂的情形下。因此,大多数多目标蚁群算法利用多个信息素矩阵和启发信息矩阵。一般的做法是,每一个目标都有一个信息素矩阵和启发信息矩阵,这能体现相对于某个特定的目标而言解元素的作用。不过,这种方法不利于处理更为复杂的问题。为了更好地逼近 Pareto 前沿,有必要提出一种有效的方法来确定合适的信息素矩阵和启发信息矩阵数目以记录有用的信息。

(2) 解构造:多目标蚁群算法在每一代应该能构造一组多样化的新解以更好地逼近整个 Pareto 前沿。为此,在解构造过程中,不同的蚂蚁利用不同的信息素和启发信息组合,从而能向 Pareto 前沿不同部分逼近。如何合理地组合信息素和启发信息是本章的重点内容之一。

基于分解的多目标进化算法是一种求解多目标问题的进化算法框架[10]。与 MOGLS、MSOPS 和 AWA 等多目标进化算法类似[10,11],它也是基于经典的集结方法:一个多目标问题被分解成一组单目标优化子问题。每一个子问题的目标是多个目标的一个线性或非线性加权组合。子问题的邻居关系依据子问题对应的权向量之间的距离来确定。每个子问题利用在求解邻居子问题过程中获取的信息来进行优化。基于分解的多目标进化算法已经用于求解许多多目标优化问题。

本章在基于分解的多目标进化算法框架下提出一种多目标蚁群算法。在设计该算法时,特别注意到上述两个问题。在该算法中,每个蚂蚁负责求解一个子问题并且针对 Pareto 前沿的一部分。每个蚂蚁有一个启发信息矩阵用以记录其对应子问题的先验信息。依据子问题的邻域结构,这些蚂蚁被分成若干个群。每个群逼近 Pareto 前沿的特定部分。在一个群中的蚂蚁共享一个信息素矩阵,它用以记录 Pareto 子区域的习得信息。在搜索过程中,每个蚂蚁记录它为所负责的子问题找到的当前最优解。在构造一个新解时,蚂蚁融合它所在群的信息素矩阵信息和自身的启发信息以及当前最优解信息。以这种方式构造的矩阵再用以引导解构造过程。在这种方

式下，构造过程更可能得到蚂蚁所对应子问题的好解。每个子群的信息素矩阵利用它所对应的蚂蚁构造的新解来更新。基于每个蚂蚁对应的目标函数，如果它能从其邻域中得到新解，蚂蚁更新其当前解。这使不同群的蚂蚁之间也可以相互合作，这是因为两个相邻的蚂蚁可以属于不同的群。

BiCriterionAnt[12]是一类最早利用不同群的蚂蚁来搜索 Pareto 前沿不同区域的多目标蚁群算法。本章提出的算法与 BiCriterionAnt 算法的区别在于：①每个蚂蚁记录子问题的一个解，且该解直接作用于解结构。从一定意义上讲，它融合了个体局部信息和全局信息。而 BiCriterionAnt 没有这种机制。②本算法依赖于个体之间的邻域关系，使不同的蚂蚁子群之间能协作。而 BiCriterionAnt 中不同群之间通过一个全局解池来共享信息。③本算法中，蚂蚁仅仅更新自己子群的信息素矩阵，这样计算量小而且容易计算。

10.2 多目标优化的基本概念

一般地，多目标优化算法可以表述为

$$\min F(x) = (f_1(x), \cdots, f_m(x))$$
$$\text{s.t.} \quad x \in \Omega \tag{10-1}$$

其中 Ω 是变量空间，$F:\Omega \to R^m$ 由 m 个实值函数构成，R^m 称为目标空间。可达目标集定义为 $\{F(x) \mid x \in \Omega\}$。

令 $u, v \in R^m$，u 称为占优 v 当且仅当对每一个指标 $i \in \{1,2,\cdots,m\}$，都有 $u_i \leqslant v_i$，且至少有一个指标 $j \in \{1,2,\cdots,m\}$ 使得 $u_j < v_j$。Ω 中的一个点 x^* 称为 Pareto 最优，如果没有其他点 x 使得 $F(x)$ 占优 $F(x^*)$，则称 x^* 为 Pareto 最优目标向量。即对 Pareto 最优点，任何一个目标上的改进至少导致其他一个目标上的下降。所有 Pareto 最优解构成 Pareto 集，且所有 Pareto 最优目标向量称为 Pareto 前沿。

1）多目标 0-1 背包问题

给定 n 个物品，m 个背包，多目标 0-1 背包问题可定义为

$$\max f_i(x) = \sum_{j=1}^{n} p_{ij} x_j, i = 1,2,\cdots,m$$
$$\text{s.t.} \quad \sum_{j=1}^{n} w_{ij} x_j \leqslant c_i, i = 1,2,\cdots,m$$
$$x = (x_1, x_2, \cdots, x_n) \in \{0,1\}^n \tag{10-2}$$

其中 $p_{ij} \geqslant 0$ 是物品 j 放到背包 i 的收益，$w_{ij} \geqslant 0$ 是物品 j 放到背包 i 的代

价，c_i 是背包 i 的总容量。$x_j=1$ 表示物品 j 被选择放入所有的背包。

2）多目标旅行商问题

给定 n 个城市和 m 个距离矩阵 $(c_{jk}^i)_{n\times n}$，$i=1,2,\cdots,m$，$c_{jk}^i>0$ 是从城市 j 到 k 的代价。这个问题可以描述为

$$\min f_i(x)=\sum_{j=1}^{n-1}c_{x_jx_{j+1}}^i+c_{x_1x_n}^i$$
$$i=1,2,\cdots,m \tag{10-3}$$

其中 x 是 n 个城市的一个全排列，表示一条经过每个城市一次且仅一次的路线。

在数学规划文献中已经有多种将一个多目标优化问题分解成一组单目标优化问题的方法。下面介绍两种重要的分解方法。

加权求和法：令 $\boldsymbol{\lambda}=(\lambda_1,\lambda_2,\cdots,\lambda_m)$ 是权重向量，且 $\sum_{i=1}^{m}\lambda_i=1,\lambda_i\geqslant 0$，$i=1,2,\cdots,m$。如果 Pareto 前沿是凸的，则下列优化问题的最优解是(10.1)的 Pareto 最优解。

$$\min g(x\mid\boldsymbol{\lambda})=\sum_{i=1}^{m}\lambda_i f_i(x)$$
$$\text{s.t.}\quad x\in\Omega \tag{10-4}$$

其中 $g(x|\boldsymbol{\lambda})$ 是具有权重 $\boldsymbol{\lambda}=(\lambda_1,\lambda_2,\cdots,\lambda_m)$ 的加权目标函数。如果 Pareto 前沿是非凸的，则通过求解加权求和子问题可能得不到一些 Pareto 最优解。

Tchebycheff 法：子问题的目标函数定义为

$$\min g(x\mid\boldsymbol{\lambda})=\max_{1\leqslant i\leqslant m}\{\lambda_i(f_i(x)-z_i^*)\}$$
$$\text{s.t.}\quad x\in\Omega \tag{10-5}$$

其中 $z^*=(z_1^*,z_2^*,\cdots,z_m^*)$ 是参考点，$z_i^*=\max_{x\in\Omega}f_i(x)$。

10.3 基于分解的多目标蚁群算法

MOEA/D-ACO 将一个多目标问题分解成 N 个单目标子问题。每个子问题 i 对应的权重向量为 $\boldsymbol{\lambda}^i$，其对应的子问题记为 $g(x|\boldsymbol{\lambda}^i)$。MOEA/D-ACO 用 N 个蚂蚁求解这些子问题，每个蚂蚁分别负责一个子问题，即蚂蚁 i 负责子问题 $g(x|\boldsymbol{\lambda}^i)$。

由于 $g(x|\boldsymbol{\lambda})$ 是 $\boldsymbol{\lambda}$ 连续函数，如果两个子问题的权重距离小，则这两个问题的解相似程度高。受此启发，MOEA/D-ACO 采用如下概念。

(1) 邻域：蚂蚁 i 的邻域 $B(i)$ 包含 T 个最相邻的蚂蚁，即这些蚂蚁的

权重向量与蚂蚁 i 的权重向量距离比其他蚂蚁对应的权重向量到蚂蚁 i 的距离都小。显然有 $i \in B(i)$。

(2) 子群：N 个蚂蚁依据其权重向量的聚类被分成 K 个子群。每个子群负责逼近 Pareto 面的一部分。

图 10-1 给出了 MOEA/D-ACO 中分解、邻域和子群的概念。

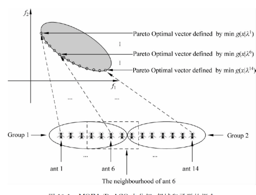


图 10-1　MOEA/D-ACO 中分解、邻域和子群的概念

图 10-1 给出的实例中，目标数(m)为 2，蚂蚁数(N)为 14，子群数(K)为 2，邻域大小(T)为 5。该多目标优化问题被分解成 14 个单目标子问题。蚂蚁 i 负责优化子问题 i。第一个子群由蚂蚁 1 到 7 组成，其他蚂蚁构成第二个子群。每个蚂蚁有 5 个邻居，如蚂蚁 6 的邻居为蚂蚁 4、5、6、7 和 8。

在每一代，种群规模为 N 的 MOEA/D-ACO 维护以下信息。

(1) N 个解 $\boldsymbol{x}^1, \boldsymbol{x}^2 \cdots, \boldsymbol{x}^N \in \Omega$，其中 $\boldsymbol{x}^i$ 是子问题 i 的当前解。

(2) $F^1, F^2, \cdots, F^N$，其中 F^i 是 x^i 的目标函数值(基于子问题 i 的目标函数)，即 $F^i = F(\boldsymbol{x}^i)$ 。

(3) $\boldsymbol{\tau}^1, \boldsymbol{\tau}^2, \cdots, \boldsymbol{\tau}^K$，其中 $\boldsymbol{\tau}^j$ 是子群 j 的信息素矩阵，用以保存子群 j 习得信息。

(4) $\boldsymbol{\eta}^1, \boldsymbol{\eta}^2, \cdots, \boldsymbol{\eta}^N$，其中 $\boldsymbol{\eta}^i$ 是子问题 i 对应的启发信息矩阵。

(5) EP，一个外部档案集，用以保存现有非占优解的目标函数值向量。

MOEA/D-ACO 主要步骤如下。

步骤 0(初始化)：对 $i=1,2,\cdots,N$,对每个子问题 i 产生一个初始解,令 $F^i=F(\boldsymbol{x}^i)$；并初始化其信息素矩阵。对 $j=1,2,\cdots,K$,初始化每个子群 j 的信息素矩阵 $\boldsymbol{\tau}^j$。初始化 EP 为 $\{F^1,\cdots,F^N\}$；

步骤 1(解构造)：对 $i=1,2,\cdots,N$,蚂蚁 i 利用概率规则选择解元素来构造一个新解 y^i。这个规则依赖于 x^i,$\boldsymbol{\eta}^i$,$\boldsymbol{\tau}^j$,其中对 $i=1,2,\cdots,N$,子问题 i 在子群 j 中。计算 $F(\boldsymbol{y}^i)$；

步骤 2(EP 的更新)：对 $i=1,2,\cdots,N$,如果 EP 中没有向量占优 $F(\boldsymbol{y}^i)$,则将其加入到 EP 中,并且将 EP 中那些被 $F(\boldsymbol{y}^i)$ 占优的向量移除；

步骤 3(终止)：如果算法终止条件满足,则停止算法,并输出 EP；

步骤 4(信息素矩阵的更新)：对 $j=1,2,\cdots,K$,利用子群 j 中蚂蚁构造的新解以及 EP 中的新解来更新信息素矩阵 $\boldsymbol{\tau}^j$；

步骤 5 (x^i 的更新)：对 $i=1,2,\cdots,N$,假设蚂蚁 i 找到的新解为 $\boldsymbol{y}$,如果 $g(\boldsymbol{y}|\boldsymbol{\lambda}^i)<g(\boldsymbol{x}^i|\boldsymbol{\lambda}^i)$,则用 $\boldsymbol{y}$ 替换 $\boldsymbol{x}^i$,转入步骤 1。

在每一代中,步骤 1 是每个蚂蚁构造一个解,步骤 2 是用这些新解来更新 EP。步骤 4 是一个蚂蚁如果得到新的非占优解,则用其更新蚂蚁对应子群的信息素矩阵。步骤 5 是更新当前解。而步骤 3 是算法终止条件判断。

10.3.1 MOEA/D-ACO 求解 MOKP

1) 基本设置

(1) N 和 $\boldsymbol{\lambda}^1,\cdots,\boldsymbol{\lambda}^N$ 的设置：N 受到一个参数 H 控制。具体而言,$\boldsymbol{\lambda}^1,\cdots,\boldsymbol{\lambda}^N$ 的每个分量从以下集合中选取一个值：$\left\{0,\frac{1}{H},\cdots,\frac{H-1}{H},1\right\}$,因此共有 $N=C_{H+m-1}^{m-1}$ 种权重向量组合。

(2) 子群数的设置：采用上述类似的方法来设置 K 及每个子群的权重向量 $\boldsymbol{\xi}^1,\boldsymbol{\xi}^2,\cdots,\boldsymbol{\xi}^K$,再对 $\boldsymbol{\lambda}^1,\boldsymbol{\lambda}^2,\cdots,\boldsymbol{\lambda}^N$ 分别计算到 $\boldsymbol{\xi}^1,\boldsymbol{\xi}^2,\cdots,\boldsymbol{\xi}^K$ 距离最近的一个向量,从而给每个蚂蚁指定子群号。

(3) 邻域的设置：欧氏距离用以计算权重向量的邻域。

(4) 启发信息和信息素：在多目标背包问题中,每个解被编码成 0-1 向量。采用 n 维实值向量来表征信息素和启发信息。具体而言,子群 j 的信息素矩阵为

$$\boldsymbol{\tau}^j=(\tau_1^j,\tau_2^j,\cdots,\tau_n^j) \tag{10-6}$$

其中 τ_k^j 表征在以前搜索中习得的物品 k 被子群 j 选择的价值。蚂蚁 i 的启发信息矩阵定义为

$$\boldsymbol{\eta}^i=(\eta_1^i,\eta_2^i,\cdots,\eta_n^i) \tag{10-7}$$

其中,η_k^i 表征物品 k 被蚂蚁 i 的选择的先验价值。

2）初始化

启发信息向量$\boldsymbol{\eta}^i$的第k个分量取值为

$$\eta_k^i = \frac{\sum_{l=1}^{m} \lambda_l^i p_{lk}}{\sum_{l=1}^{m} \gamma_l^i w_{lk}} \tag{10-8}$$

其中γ_l^i为线性规划松弛问题的第l个约束的影子价格。

所有信息素向量的分量都初始化为一个相同的较大值，以增强算法的探索能力：

$$\tau_k^i = 1 \tag{10-9}$$

3）解构造过程

假设蚂蚁i在子群j中，其当前解为$\boldsymbol{x}^i=(x_1^i, x_2^i\cdots, x_n^i)$。在算法步骤1中蚂蚁$i$以如下方式构造一个新解$\boldsymbol{y}^i=(y_1^i, y_2^i, \cdots, y_n^i)$：

对$k=1,2,\cdots,n$，记

$$\phi_k = (\tau_k^i + \Delta \times x_k^i)^\alpha (\eta_k^i)^\beta, \tag{10-10}$$

其中Δ,α,β是3个正参数。ϕ_k表征$y_k^i=1$的偏好度，即物品k被选择的偏好度。

将$\boldsymbol{y}^i$初始化为$(0,\cdots,0)$，即背包都是空的，记$I=\{1,2,\cdots,n\}$，如果$I\neq\phi$，重复以下操作。

（1）对每个$s\in I$，如果添加s到背包中导致不可行解，则将其从I中删除。

（2）如果$I=\varnothing$，则返回$\boldsymbol{y}^i$。

从$(0,1)$之间产生随机数，如果它大于参数r，则将s置为I中ϕ值最大的元素，否则按照下式依据轮盘赌规则随机选择一个元素：

$$\frac{\phi_s}{\sum_{k\in I} \phi_k} \tag{10-11}$$

将s从I中移除，并置$y_s^i=1$。

在式(10-11)中，τ_k^i是子群j中所有蚂蚁共享的信息素矩阵的第k个分量。$x_k^i=1$表示物品k在子问题i当前解中被选择。x_k^i是蚂蚁i私有信息。因此$\tau_k^i+\Delta\times x_k^i$融合了子群和私有信息。这种组合可以看成是蚂蚁$i$在物品$k$上的信息素。

在构造解过程中，s依据拟随机比例规则进行选择。参数r调节开发和探索。如果随机数小于r，则进行开发并选择偏好值最大的物品，否则进行探索搜索，此时，偏好值较大的物品被选择的可能性大。

4）信息素矩阵的更新

记 $\prod$ 为满足如下条件的新解：

（1）在当前代中被子群 j 构造出的当前解；

（2）在当前代被加到 EP 中的解；

（3）第 k 个元素为 1。

子群 j 中的第 k 个物品对应的信息素分量 τ_k^j 更新如下：

$$\tau_k^j := \rho\tau_k^j + \sum_{x\in\prod} \frac{1}{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij} - g(x \mid \lambda^j)} \tag{10-12}$$

其中，参数 ρ 是信息素的保持率。在这个信息素更新公式中，信息素矩阵保存了子群 j 中迄今为止的好解的统计信息。

为了避免有些信息素值过分大或小，信息素的上下界 $\tau_{\max}$，$\tau_{\min}$用以控制信息素的值。

依据 MMAS，在 MOEA/D-ACO 中，信息素的上界 $\tau_{\max}$在每个代更新如下：

$$\tau_{\max} = \frac{B+1}{(1-\rho)\left(\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij} - g_{\max}\right)} \tag{10-13}$$

其中，B 是当前代中非占优解的个数，$g_{\max}$是所有子问题的最大目标函数值，信息素的下界 $\tau_{\min}$置为

$$\tau_{\min} = \varepsilon\tau_{\max} \tag{10-14}$$

其中，$\varepsilon\in(0,1)$ 是一个参数。如果 τ_k^j 比 $\tau_{\min}$小，则将其赋值为 $\tau_{\min}$；如果 τ_k^j 比 $\tau_{\max}$大，则将其赋值为 $\tau_{\max}$。

10.3.2 MOEA/D-ACO 求解 MTSP

1）信息素和启发信息定义

在求解 MTSP 时，信息素和启发信息矩阵定义如下：

（1）每个子群 j 都有一个信息素矩阵，其中每个元素 τ_{kl} 表示城市 k 和 l 之间的信息素值。

（2）每个蚂蚁 i 都有一个启发信息矩阵，其中每个元素 η_{kl}^i 表示城市 k 和 l 之间的启发信息值。

（3）每个解都是城市的一个排列。

2）初始化

启发信息值初始化如下：

$$\eta_{kl}^i = \frac{1}{\sum_{j=1}^{m}\lambda_j^i c_{kl}^j} \tag{10-15}$$

而信息素 τ_{kl}^{j} 初始化为 1。

3）解构造

假设蚂蚁 i 在子群 j 中，其当前解为 $x^{i}=(x_1^{i},x_2^{i},\cdots,x_n^{i})$。在算法步骤 1 中蚂蚁 i 以如下方式构造一个新解 $y^{i}=(y_1^{i},y_2^{i},\cdots,y_n^{i})$。

对 $k=1,2,\cdots,n$，记

$$\phi_{kl}=(\tau_{kl}^{j}+\Delta\times In(x^{i},(k,l)))^{\alpha}(\eta_{kl}^{j})^{\beta}, \tag{10-16}$$

其中 Δ,α,β 是 3 个正参数。ϕ_{kl} 表征连接城市 k 和 l 之间的边的偏好度。当解 x^{i} 经过边 (k,l) 时，示性函数 $In(x^{i},(k,l))$ 为 1，否则为 0。其中，示性函数 $In(y)$ 表示 y 为 1 时，$In(y)$ 为 1；否则为 0。

蚂蚁 i 随机选择一个城市作为初始城市开始构造解。假设当前城市为 l 且还有城市未被访问，记这些城市为集 C。如果 $C\neq\varnothing$，重复以下操作。

（1）从 $(0,1)$ 之间产生随机数，如果它大于参数 r，则将 k 置为 I 中 ϕ 值最大的元素，否则按照下式依据轮盘赌规则随机选择一个元素：$\dfrac{\phi_{kl}}{\sum\limits_{s\in C}\phi_{sl}}$。

（2）将 k 从 C 中移除。

（3）如果 C 为空集，则得到解。

4）信息素矩阵的更新

记 $\prod$ 为满足如下条件的新解：

（1）在当前代中被子群 j 构造出的当前解；

（2）在当前代被加到 EP 中的解；

（3）经过边 (k,l)。

子群 j 中的边 (k,l) 对应的信息素值 τ_{kl}^{j} 更新为

$$\tau_{kl}^{j}:=\rho\tau_{kl}^{j}+\sum_{x\in\prod}\frac{1}{g(x\mid\lambda^{j})} \tag{10-17}$$

其中，参数 ρ 是信息素的保持率。

信息素的上界 $\tau_{\max}$ 在每个代更新为

$$\tau_{\max}=\frac{B+1}{(1-\rho)g_{\min}} \tag{10-18}$$

其中，B 是当前代中非占优解的个数，$g_{\min}$ 是所有子问题的最小目标函数值，信息素的下界 $\tau_{\min}$ 置为

$$\tau_{\min}=\varepsilon\tau_{\max} \tag{10-19}$$

其中，$\varepsilon\in(0,1)$ 是一个参数。如果 τ_{kl}^{j} 比 $\tau_{\min}$ 小，则将其赋值为 $\tau_{\min}$；如果 τ_{kl}^{j} 比 $\tau_{\max}$ 大，则将其赋值为 $\tau_{\max}$。

10.4 与 MOEA/D-GA 在 MOKP 上的比较

文献[10]将 MOEA/D 结合传统的遗传算子和局部搜索用以解决 MOKP 问题。其交叉算子是单点交叉，变异是将解的一个基因位以较小的概率反转。局部搜索算子采用文献[13]。

同 MOEA/D-GA 进行比较的原因是：它是一类有效的多目标进化算法，它比 MOGLS 等在内的算法性能都好，而且是基于 MOEA/D 框架。

10.4.1 实验条件

实验数据是文献[14]中的 9 个算例。

MOEA/D-GA 的参数设置如文献[10]所述。由于加权求和法比 Tchebycheff 法的结果好，仅考虑加权求和法。

为了公平比较，两个算法中相同的参数都取相同的值。在实验中，蚂蚁数和子群数见表 10-1。其他参数设置如下：$T=10$，$\alpha=1$，$\beta=10$，$\rho=0.95$，$\gamma=0.9$，$\varepsilon=\frac{1}{2n}$，$\Delta=0.05\tau_{\max}$。

表 10-1　MOEA/D-ACO 的参数设置

Instance		$N(H)$	$K(H)$
n	m		
250	2	150 (149)	9 (8)
500	2	200 (199)	9 (8)
750	2	250 (249)	9 (8)
250	3	351 (25)	10 (3)
500	3	351 (25)	10 (3)
750	3	351 (25)	10 (3)
250	4	455 (12)	20 (3)
500	4	455 (12)	20 (3)
750	4	455 (12)	20 (3)

所有算法在 300 代后终止。所有的实验都在相同的计算机上测试。

10.4.2 性能评价指标

采用 IGD(inverted generational distance)来评价性能。

令 P^* 为一组沿着 Pareto 前沿均匀分布的点组成的集合；P 是 Pareto 前沿的一个逼近。则 P^* 和 P 之间的 IGD 定义为

$$D(P^*,P)=\frac{\sum_{v\in P^*}d(v,P)}{|P^*|} \tag{10-20}$$

其中,$d(v,P)$是点 v 和集合 P 之间的距离,即点 v 到集合 P 中点的最小距离。如果 P^* 足够大,能足够好表示 Pareto 前沿,则 $D(P^*,P)$能刻画多样性和收敛效果。为得到一个小的 $D(P^*,P)$,P 必须很好地逼近 Pareto 前沿,且不能遗漏其中任何部分。由于真实的 Pareto 前沿不能得到,在实验中,P^* 为文献[13]中的上逼近。

10.4.3 结果比较

将每个算法在每个算例测试 30 次,得到 30 个非占优面。

表 10-2 给出了这些非占优面的 IGD 的平均值和标准差。表 10-3 给出了每个算法的 CPU 时间开销。图 10-2 给出了每个算法 30 次中最小 IGD(即最佳 IGD)非占优面的分布图。MOEA/D-ACO^W 和 MOEA/D-ACO^T 分别表示采用加强求和法和 Tchebycheff 法进行分解的 MOEA/D-ACO。图 10-3～图 10-5 给出了平均 IGD 值随着迭代步数变化的曲线。

表 10-2 MOEA/D-ACO^W、MOEA/D-ACO^T 以及 MOEA/D-GA 的 IGD 值比较

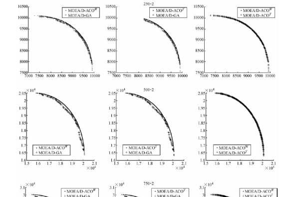
Instance		MOEA/D-ACO^W			MOEA/D-ACO^T			MOEA/D-GA	
n	m	mean	std dev	t-test	mean	std dev	t-test	mean	std dev
250	2	13.8	0.3	+	13.4	0.4	+	59.1	6.3
500	2	12.7	0.5	+	12.7	0.6	+	157.1	16.5
750	2	14.0	0.5	+	14.4	0.5	+	401.7	36.4
250	3	46.6	1.4	+	47.3	1.5	+	119.7	8.2
500	3	73.9	2.6	+	77.2	2.3	+	357.1	24.3
750	3	101.2	3.1	+	106.1	3.9	+	696.1	30.7
250	4	90.3	1.4	+	88.7	1.2	+	195.2	6.3
500	4	175.4	1.5	+	170.3	1.9	+	534.0	15.2
750	4	280.7	1.9	+	270.8	2.1	+	1064.5	28.1

表 10-3 MOEA/D-ACO^W、MOEA/D-ACO^T 以及 MOEA/D-GA 的计算时间比较

Instance		MOEA/D-ACO^W		MOEA/D-ACO^T		MOEA/D-GA	
n	m	mean	std dev	mean	std dev	mean	std dev
250	2	6.4	0.0	6.2	0.0	0.9	0.0
500	2	10.9	0.0	10.7	0.0	2.8	0.0
750	2	15.7	0.0	15.5	0.1	5.8	0.1
250	3	15.9	0.9	16.2	0.0	2.3	0.1

续表

Instance		MOEA/D-ACO^W		MOEA/D-ACO^T		MOEA/D-GA	
n	m	mean	std dev	mean	std dev	mean	std dev
500	3	24.6	0.1	24.3	0.0	7.2	0.2
750	3	33.7	0.1	34.0	0.1	13.1	0.3
250	4	26.1	0.0	25.3	0.5	6.4	0.1
500	4	43.1	0.0	42.3	0.0	16.2	0.2
750	4	54.6	0.1	55.3	0.1	29.4	0.4



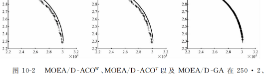


图 10-2 MOEA/D-ACO^W、MOEA/D-ACO^T 以及 MOEA/D-GA 在 250 • 2、500 • 2 和750 • 2 3 个算例上得到的 30 次实验结果中最好的逼近前沿面

从表 10-2 中可以看到依据 IGD 指标，两个版本的 MOEA/D-ACO 在所有算例上都比 MOEA/D-GA 要好。例如，在算例 250 • 2，500 • 3，750 •

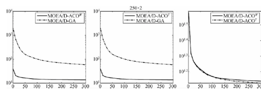


图 10-3　在算例 250・2 上的 IGD 演化曲线

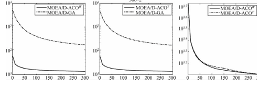


图 10-4　在算例 500・2 上的 IGD 演化曲线

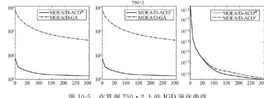


图 10-5　在算例 750・2 上的 IGD 演化曲线

4 上，MOEA/D-ACO 分别改进了 23%，22%，25%。从图 10-2 中可以看到，两个版本的 MOEA/D-ACO 差异很小基本上难以分辨，基于加权求和法的 MOEA/D-ACO 略好。

从图 10-2 中可见，在 250・2，500・2，750・2 3 个算例上，基本上所有 MOEA/D-GA 得到的非占优面都被占优于 MOEA/D-ACO 得到的非占优面。随着决策变量的增加，MOEA/D-GA 和 MOEA/D-ACO 的性能差异越大。表 10-3 表明 MOEA/D-ACO 花费的时间略多于 MOEA/D-GA。这是因为 ACO 的计算开销要大于交叉算子和变异算子。然而，如果以构造解

的个数来看,图 10-3～图 10-5 表明 MOEA/D-ACO 更为有效。

MOEA/D-GA 和 MOEA/D-ACO 都建立在 MOEA/D 框架下。因此,MOEA/D-ACO 的优势在于 ACO 部分。ACO 的一个优势在于它很好地利用到问题的相关启发信息来进行构造解。从式(10-10)中可见,当 $\beta=0$ 时,MOEA/D-ACO 将不利用启发信息。为了研究启发信息对应算法性能的影响,考虑 $\beta=0$ 且其他参数取值都与前所述时得到的结果。表 10-4 给出了所有算例上的实验结果。可以很清楚地看到,如果没有启发信息,MOEA/D-ACO 结果很差。以算例 250・2 为例子,MOEA/D-ACOW 没有启发信息时,平均 IGD 是有启发信息时的 79 倍。可以看到,如果没有启发信息,MOEA/D-ACO 比 MOEA/D-GA 都差。因此,有效利用启发信息对于算法非常关键。

表 10-4　MOEA/D-ACO^W 与 MOEA/D-ACO^T 在 β 取值不同时的 IGD 值比较

Instance		MOEA/D-ACO^W					MOEA/D-ACO^T				
		default setting		$\beta=0$			default setting		$\beta=0$		
n	m	mean	std dev	mean	std dev	t-test	mean	std dev	mean	std dev	t-test
250	2	13.8	0.3	1101.9	137.3	+	13.4	0.4	1030.5	142.0	+
500	2	12.7	0.5	3728.9	349.8	+	12.7	0.6	3690.5	428.0	+
750	2	14.0	0.5	14.4	658.7	+	14.4	0.5	5171.4	688.6	+
250	3	46.6	1.4	47.3	28.4	+	47.3	1.5	803.5	92.1	+
500	3	73.9	2.6	77.2	287.3	+	77.2	2.3	2847.5	295.4	+
750	3	101.2	3.1	106.1	865.6	+	106.1	3.9	4803.9	580.9	+
250	4	90.3	1.4	88.7	38.0	+	88.7	1.2	857.7	82.1	+
500	4	175.4	1.5	170.3	89.7	+	170.3	1.9	2619.4	140.1	+
750	4	280.7	1.9	270.8	381.8	+	270.8	2.1	4104.4	418.3	+

10.5　与 BicriterionAnt 在 MTSP 上的比较

BicriterionAnt 是最好的多目标蚁群算法之一,而且它也用到子群的概念。通过比较 MOEA/D-ACO 算法和 BicriterionAnt 算法,可以分析分解方法的作用。

10.5.1　实验条件

采用 TSP 网站 http://eden.dei.uc.pt/～paquete/tsp/中的算例用于实验分析。

参数设置如下：

子群数目为 3，每个子群中蚂蚁数为 8，$\rho=0.95$，$\alpha=1$，$\beta=2$，$\gamma=0.9$，邻域的大小 $T=10$，$\epsilon=\frac{1}{2n}$，$\Delta=0.05\tau_{\max}$。

在计算 IGD 值时，P^* 为实验中得到的非占优解(即每次得到的非占优解之后，再得到这些解的非占优解)。每个算例都测试 30 遍。

10.5.2 实验结果

表 10-5 给出了 MOEA/D-ACO^W、MOEA/D-ACO^T 和 BicriterionAnt 的非占优面的 IGD 的平均和标准差。可见，这 3 个算法的结果都比较接近，在一个数值精度下。从图 10-6 可见，3 个算法的差异是明显的。MOEA/D-ACO^W、MOEA/D-ACO^T 几乎完全优于 BicriterionAnt。图 10-7～图 10-9 表明 MOEA/D-ACO 能有效减小 IGD 值。实验结果表明分解方法是有效的。而且，在表 10-6 中可见 MOEA/D-ACO^W、MOEA/D-ACO^T 比 BicriterionAnt 要花费更少的时间，这是因为后者要花许多时间进行解的比较。

表 10-5 MOEA/D-ACO^W、MOEA/D-ACO^T 以及 BicriterionAnt 的 IGD 值比较

Instance		MOEA/D-ACO^W			MOEA/D-ACO^T			BicriterionAnt	
name	*n*	mean	std dev	t-test	mean	std dev	t-test	mean	std dev
kroab100	100	2008.2	330.3	+	1865.7	318.6	+	12318.5	382.9
kroac100	100	1733.0	254.9	+	1697.8	288.4	+	13209.0	373.1
kroad100	100	1494.6	217.3	+	1440.2	200.8	+	11888.0	403.1
kroae100	100	1887.3	303.5	+	1669.6	290.3	+	12526.1	368.9
krobc100	100	1802.5	345.3	+	1452.9	172.8	+	13881.1	387.9
krobd100	100	1777.3	298.3	+	1655.1	215.7	+	13625.8	271.0
krobe100	100	2285.9	292.8	+	1864.9	301.8	+	15042.7	458.3
krocd100	100	1897.3	295.4	+	1650.2	299.4	+	11688.2	274.0
kroce100	100	1852.2	311.2	+	1710.3	275.9	+	13050.4	324.7
krode100	100	2017.1	340.5	+	1904.6	375.7	+	12451.9	328.1
kroab200	200	4476.7	598.6	+	3910.1	393.8	+	32691.2	908.1
kroab300	300	8735.9	1121.2	+	6441.4	595.2	+	51776.2	1171.5

表 10-6 MOEA/D-ACO^W、MOEA/D-ACO^T 以及 BicriterionAnt 的 IGD 值比较

Instance		MOEA/D-ACO^W		MOEA/D-ACO^T		BicriterionAnt	
name	*n*	mean	std dev	mean	std dev	mean	std dev
kroab100	100	4.9	0.1	4.9	0.1	68.4	0.1
kroac100	100	4.9	0.1	4.9	0.1	68.3	0.0
kroad100	100	4.9	0.1	4.9	0.1	68.3	0.0
kroae100	100	4.9	0.1	4.9	0.0	68.3	0.1
krobc100	100	5.0	0.0	4.9	0.1	68.6	0.2

续表

Instance		MOEA/D-ACO^W		MOEA/D-ACO^T		BicriterionAnt	
name	*n*	mean	std dev	mean	std dev	mean	std dev
krobd100	100	4.9	0.1	4.9	0.0	68.8	0.1
krobe100	100	4.9	0.1	4.9	0.0	68.8	0.1
krocd100	100	4.9	0.0	4.9	0.1	68.7	0.0
kroce100	100	4.9	0.0	4.9	0.0	68.6	0.1
krode100	100	4.9	0.0	4.9	0.0	68.5	0.1
kroab200	200	20.4	0.2	20.3	0.1	276.3	0.3
kroab300	300	45.1	0.5	45.1	0.4	629.1	1.1

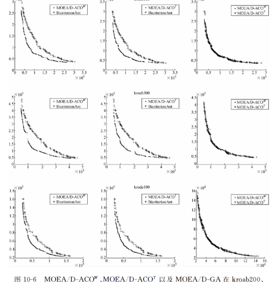


图 10-6　MOEA/D-ACO^W、MOEA/D-ACO^T 以及 MOEA/D-GA 在 kroab200、kroab300 和 krode100 3 个算例上得到的 30 次实验结果中最好的逼近前沿面

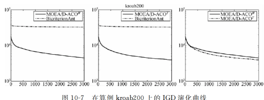


图 10-7　在算例 kroab200 上的 IGD 演化曲线

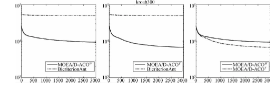


图 10-8　在算例 kroab300 上的 IGD 演化曲线

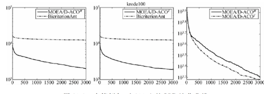


图 10-9　在算例 krode100 上的 IGD 演化曲线

MOEA/D-ACO 和 BicriterionAnt 都用 ACO 来构造解。其差异在于 MOEA/D-ACO 采用当前解用于解构造，而且利用到子群和邻域的概念来交换信息。下面来分析以下 3 个问题：

（1）如果不用子群，效果如何？即考虑 $k=1$ 时算法性能。

（2）如果不用当前解，效果如何？即考虑 $\Delta=0$ 时算法性能。

（3）如果相邻子问题之间不交换信息，效果如何？即考虑 $T=1$ 时算法性能。

实验结果在表 10-7 和表 10-8 中给出。由结果可见：

表 10-7 MOEA/D-ACO^W 在不同参数情形下的 IGD 值比较

Instance		default setting		$K=1$			$\Delta=0$			$T=1$		
name	*n*	mean	std dev	mean	std dev	t-test	mean	std dev	t-test	mean	std dev	t-test
kroab100	100	1865.7	318.6	3217.1	592.1	+	2379.6	298.5	+	1963.6	408.1	+
kroac100	100	1697.8	288.4	3034.1	547.3	+	2125.7	281.2	+	1789.1	320.5	+
kroad100	100	1440.2	200.8	2356.6	317.9	+	1996.7	216.6	+	1592.4	187.8	+
kroae100	100	1669.6	290.3	2826.3	834.3	+	2142.0	330.8	+	1740.6	261.1	+
krobc100	100	1452.9	172.8	2544.2	336.4	+	2080.6	198.0	+	1756.4	289.3	+
krobd100	100	1655.1	215.7	2770.1	312.1	+	2133.9	222.8	+	1790.7	216.9	+
krobe100	100	1864.9	301.8	2963.9	542.6	+	2431.6	209.0	+	1976.8	266.0	+
krocd100	100	1650.2	299.4	2568.1	464.5	+	2074.6	289.9	+	1691.0	243.9	+
kroce100	100	1710.3	275.9	2590.6	389.2	+	2166.5	219.6	+	1925.3	307.4	+
krode100	100	1904.6	375.7	2714.3	487.0	+	2255.5	308.2	+	1947.8	324.6	+
kroab200	200	3910.1	393.8	5465.2	784.2	+	5487.8	545.2	+	4091.1	373.3	+
kroab300	300	6441.4	595.2	7768.9	808.2	+	8249.5	679.0	+	6477.2	604.6	+

表 10-8　MOEA/D-ACO^T 在不同参数情形下的 IGD 值比较

Instance		default setting		$K=1$			$\Delta=0$			$T=1$		
name	n	mean	std dev	mean	std dev	t-test	mean	std dev	t-test	mean	std dev	t-test
kroab100	100	2008.2	330.3	2998.3	540.6	+	2417.2	294.1	+	2488.5	404.8	+
kroac100	100	1733.0	254.9	2916.1	477.7	+	2045.9	248.2	+	2045.1	276.6	+
kroad100	100	1494.6	217.3	2356.8	355.8	+	1957.4	182.8	+	1855.2	252.7	+
kroae100	100	1887.3	303.5	2740.2	317.4	+	2051.8	285.2	+	2042.2	329.7	+
krobc100	100	1802.5	345.3	2617.2	486.3	+	2011.1	230.2	+	2141.9	307.8	+
krobd100	100	1777.3	298.3	2601.9	407.8	+	2175.3	216.2	+	2247.6	238.5	+
krobe100	100	2285.9	292.8	3015.1	641.5	+	2371.3	250.4	+	2723.2	352.7	+
krocd100	100	1897.3	295.4	2538.3	401.9	+	2081.9	299.8	+	2216.2	344.0	+
kroce100	100	1852.2	311.2	2800.4	577.2	+	2134.8	218.2	+	2283.2	291.3	+
krode100	100	2017.1	340.5	2835.9	554.3	+	2241.6	221.9	+	2308.8	274.0	+
kroab200	200	4476.7	598.6	5486.5	762.2	+	5612.1	556.3	+	5293.9	392.3	+
kroab300	300	8735.9	1121.2	8960.6	1534.7	+	9579.5	742.8	+	9580.3	901.8	+

（1）采用子群，能远好于不采用子群。例如，在算例 kroab200 中，采用子群的 MOEA/D-ACO 能减少 IGD 值 20%左右。

（2）利用当前解能远好于不利用当前解。例如，在算例 kroab100 中，MOEA/D-ACO 利用了当前解得到的 IGD 是没有当前解的 MOEA/D-ACO 得到的 IGD 值的 78%。

（3）当交换信息时，MOEA/D-ACO 在所有算例都能得到更好的结果。

10.6 小结

本章介绍了一种多目标蚁群算法。该算法将多目标问题分解成一组单目标优化问题，每个蚂蚁负责求解一个子问题。所有的蚂蚁被分成若干个群且每个蚂蚁都有几个邻近的蚂蚁。一个蚂蚁群共享一个信息素矩阵且每个蚂蚁都有一个启发信息矩阵。在搜索过程中，每个蚂蚁记录它所负责的子问题找到的当前最优解。在构造一个新解时，蚂蚁融合它所在群的信息素矩阵信息和自身的启发信息以及当前最优解信息。蚂蚁会检查自己以及其邻居构造的新解，如果有一个解比它的当前解好(基于所对应子问题的目标函数)，则更新其当前解为检验算法的性能，将其用于求解多目标背包问题和多目标旅行商问题，并与现有算法相比较。实验结果验证了所提出的算法的有效性。

参考文献

[1] K. Miettinen. Nonlinear Multiobjective Optimization[M]. Boston, MA: Kluwer, 1999.

[2] K. Deb. Multi-Objective Optimization Using Evolutionary Algorithms[M]. New York: Wiley, 2001.

[3] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, and C. Stummer. Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection[J]. Ann. Oper. Res. 2004, 131(1-4): 79～99.

[4] C. Garcia-Martinez, O. Cordon, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP[J]. Eur. J. Oper. Res. 2007, 180(1): 116～148.

[5] D. Angus and C. Woodward. Multiple objective ant colony optimization [J]. Swarm Intell. 2009, 3(1): 69～85.

[6] I. Alaya, C. Solnon, and K. Ghedira. Ant colony optimization for multiobjective optimization problems[C]. in Proc. 19th IEEE Int. Conf. Tools Artif. Intell. 2007, 450～457.

[7] M. López-Ibá?ez and T. Stützle. The impact of design choices of multiobjective ant colony optimization algorithms on performance: An experimental study on the biobjective TSP[C]. in Proc. GECCO, 2010, 71～78.

[8] M. Guntsch and M. Middendorf. A population based approach for ACO applications of evolutionary computing [C]. in Proc. Appl. Evol. Comput. EvoWorkshops—EvoCOP, EvoIASP, EvoSTIM/EvoPLAN. 2002,72～81.

[9] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi colony ant algorithms[C]. in Proc. 1st Int. Conf. EMO, 2001,359～372.

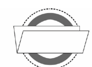
[10] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition[J]. IEEE Trans. Evol. Comput. 2007,11(6): 712～731.

[11] H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm and its application to flowshop scheduling[J]. IEEE Trans. Syst., Man, Cybern. C, Appl. Rev. 1998,28(3): 392～403.

[12] S. Iredi, D. Merkle, and M. Middendorf. Bi-criterion optimization with multi olony ant algorithms[C]. in Proc. 1st Int. Conf. EMO, 2001,359～372.

[13] A. Jaszkiewicz. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem—A comparative experiment[J]. IEEE Trans. Evol. Comput. ,2002,6(4): 402～412.

[14] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach[J]. IEEE Trans. Evol. Comput. , 1999,3(4): 257～271.



附　录

在团队定向问题中采用蚁群算法，本书第 6 章考虑了串行法、确定同步法、随机同步法和同时法共 4 种构造方法。第 6 章给出了 4 种构造方法在第 4 个和第 7 个数据集中的计算结果，本附录列出了 4 种构造方法在第 1 个、第 2 个、第 3 个、第 5 个和第 6 个数据集中的计算结果。

表附 1　4 种构造方法在第 1 个数据集中的实验结果

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p1.2.b	15	15	15	15	15	15	15	15
p1.2.c	20	20	20	20	20	20	20	20
p1.2.d	30	30	30	30	30	30	30	30
p1.2.e	45	45	45	45	45	45	45	45
p1.2.f	80	80	80	80	80	80	80	80
p1.2.g	90	90	90	90	90	90	90	90
p1.2.h	110	110	110	110	110	110	110	110
p1.2.i	135	135	135	135	135	135	135	135
p1.2.j	155	155	155	155	155	155	155	155
p1.2.k	175	175	175	175	175	175	175	175
p1.2.l	195	195	195	195	195	195	195	195
p1.2.m	215	215	215	215	215	215	215	215
p1.2.n	235	235	235	235	235	235	235	235



续表

算 例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
pl.2.o	240	240	240	240	240	240	240	240
pl.2.p	250	250	250	250	250	250	250	250
pl.2.q	265	265	265	265	265	265	265	265
pl.2.r	280	280	280	280	280	280	280	280
pl.3.c	15	15	15	15	15	15	15	15
pl.3.d	15	15	15	15	15	15	15	15
pl.3.e	30	30	30	30	30	30	30	30
pl.3.f	40	40	40	40	40	40	40	40
pl.3.g	50	50	50	50	50	50	50	50
pl.3.h	70	70	70	70	70	70	70	70
pl.3.i	105	105	105	105	105	105	105	105
pl.3.j	115	115	115	115	115	115	115	115
pl.3.k	135	135	135	135	135	135	135	135
pl.3.l	155	155	155	155	155	155	155	155
pl.3.m	175	175	175	175	175	175	175	175
pl.3.n	190	190	190	190	190	190	190	190
pl.3.o	205	205	205	205	205	205	205	205
pl.3.p	220	220	220	220	220	220	220	220
pl.3.q	230	230	230	230	230	230	230	230
pl.3.r	250	250	250	250	250	250	250	250
pl.4.d	15	15	15	15	15	15	15	15
pl.4.e	15	15	15	15	15	15	15	15
pl.4.f	25	25	25	25	25	25	25	25
pl.4.g	35	35	35	35	35	35	35	35
pl.4.h	45	45	45	45	45	45	45	45
pl.4.i	60	60	60	60	60	60	60	60
pl.4.j	75	75	75	75	75	75	75	75
pl.4.k	100	100	100	100	100	100	100	100
pl.4.l	120	120	120	120	120	120	120	120
pl.4.m	130	130	130	130	130	130	130	130
pl.4.n	155	155	155	155	155	155	155	155
pl.4.o	165	165	165	165	165	165	165	165
pl.4.p	175	175	175	175	175	175	175	175
pl.4.q	190	190	190	190	190	190	190	190
pl.4.r	210	210	210	210	210	210	210	210

表附2　4种构造方法在第2个数据集中的实验结果

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p2.2.a	90	90	90	90	90	90	90	90
p2.2.b	120	120	120	120	120	120	120	120
p2.2.c	140	140	140	140	140	140	140	140
p2.2.d	160	160	160	160	160	160	160	160
p2.2.e	190	190	190	190	190	190	190	190
p2.2.f	200	200	200	200	200	200	200	200
p2.2.g	200	200	200	200	200	200	200	200
p2.2.h	230	230	230	230	230	230	230	230
p2.2.i	230	230	230	230	230	230	230	230
p2.2.j	260	260	260	260	260	260	260	260
p2.2.k	275	275	275	275	275	275	275	275
p2.3.a	70	70	70	70	70	70	70	70
p2.3.b	70	70	70	70	70	70	70	70
p2.3.c	105	105	105	105	105	105	105	105
p2.3.d	105	105	105	105	105	105	105	105
p2.3.e	120	120	120	120	120	120	120	120
p2.3.f	120	120	120	120	120	120	120	120
p2.3.g	145	145	145	145	145	145	145	145
p2.3.h	165	165	165	165	165	165	165	165
p2.3.i	200	200	200	200	200	200	200	200
p2.3.j	200	200	200	200	200	200	200	200
p2.3.k	200	200	200	200	200	200	200	200
p2.4.a	10	10	10	10	10	10	10	10
p2.4.b	70	70	70	70	70	70	70	70
p2.4.c	70	70	70	70	70	70	70	70
p2.4.d	70	70	70	70	70	70	70	70
p2.4.e	70	70	70	70	70	70	70	70
p2.4.f	105	105	105	105	105	105	105	105
p2.4.g	105	105	105	105	105	105	105	105
p2.4.h	120	120	120	120	120	120	120	120
p2.4.i	120	120	120	120	120	120	120	120
p2.4.j	120	120	120	120	120	120	120	120
p2.4.k	180	180	180	180	180	180	180	180

表附3　4种构造方法在第3个数据集中的实验结果

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p3.2.a	90	90	90	90	90	90	90	90
p3.2.b	150	150	150	150	150	150	150	150
p3.2.c	180	180	180	180	180	180	180	180
p3.2.d	220	220	220	220	220	220	220	220
p3.2.e	260	260	260	260	260	260	260	260
p3.2.f	300	300	300	300	300	300	300	300
p3.2.g	360	360	360	360	360	360	360	360
p3.2.h	410	410	410	410	410	410	410	410
p3.2.i	460	460	460	460	460	460	460	460
p3.2.j	510	510	510	510	510	510	510	510
p3.2.k	550	550	550	550	550	550	550	550
p3.2.l	590	590	590	590	590	590	590	590
p3.2.m	620	620	620	620	620	620	620	620
p3.2.n	660	660	660	660	660	660	660	660
p3.2.o	690	690	690	690	690	690	690	690
p3.2.p	720	720	720	720	720	720	720	720
p3.2.q	760	760	760	760	760	760	760	760
p3.2.r	790	790	790	790	790	790	790	790
p3.2.s	800	800	800	800	800	800	800	800
p3.2.t	800	800	800	800	800	800	800	800
p3.3.a	30	30	30	30	30	30	30	30
p3.3.b	90	90	90	90	90	90	90	90
p3.3.c	120	120	120	120	120	120	120	120
p3.3.d	170	170	170	170	170	170	170	170
p3.3.e	200	200	200	200	200	200	200	200
p3.3.f	230	230	230	230	230	230	230	230
p3.3.g	270	270	270	270	270	270	270	270
p3.3.h	300	300	300	300	300	300	300	300
p3.3.i	330	330	330	330	330	330	330	330
p3.3.j	380	380	380	380	380	380	380	380
p3.3.k	440	440	440	440	440	440	440	440
p3.3.l	480	480	480	480	480	480	480	480
p3.3.m	520	520	520	520	520	520	520	520
p3.3.n	570	570	570	570	570	570	570	570
p3.3.o	590	590	590	590	590	590	590	590
p3.3.p	640	640	640	640	640	640	640	640

续表

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p3.3.q	680	680	680	680	680	680	680	680
p3.3.r	710	710	710	710	710	710	710	710
p3.3.s	720	720	720	720	720	720	720	720
p3.3.t	760	760	760	760	760	760	760	760
p3.4.a	20	20	20	20	20	20	20	20
p3.4.b	30	30	30	30	30	30	30	30
p3.4.c	90	90	90	90	90	90	90	90
p3.4.d	100	100	100	100	100	100	100	100
p3.4.e	140	140	140	140	140	140	140	140
p3.4.f	190	190	190	190	190	190	190	190
p3.4.g	220	220	220	220	220	220	220	220
p3.4.h	240	240	240	240	240	240	240	240
p3.4.i	270	270	270	270	270	270	100	100
p3.4.j	310	310	310	310	310	310	270	270
p3.4.k	350	350	350	350	350	350	310	310
p3.4.l	380	380	380	380	380	380	350	350
p3.4.m	390	390	390	390	390	390	390	390
p3.4.n	440	440	440	440	440	440	440	440
p3.4.o	500	500	500	500	500	500	500	500
p3.4.p	560	560	560	560	560	560	560	560
p3.4.q	560	560	560	560	560	560	560	560
p3.4.r	600	600	600	600	600	600	600	600
p3.4.s	670	670	670	670	670	670	670	670
p3.4.t	670	670	670	670	670	670	670	670

表附4　4种构造方法在第5个数据集中的实验结果

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p5.2.b	20	20	20	20	20	20	20	20
p5.2.c	50	50	50	50	50	50	50	50
p5.2.d	80	80	80	80	80	80	80	80
p5.2.e	180	180	180	180	180	180	180	180
p5.2.f	240	240	240	240	240	240	240	240
p5.2.g	320	320	320	320	320	320	320	320
p5.2.h	410	404.5	410	402.5	410	403	410	403.5

续表

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p5.2.i	480	480	480	480	480	480	480	480
p5.2.j	580	580	580	580	580	580	580	580
p5.2.k	670	670	670	669.5	670	669.5	670	670
p5.2.l	800	778	800	773	800	773	800	774
p5.2.m	860	859.5	860	859.5	860	859	860	860
p5.2.n	925	921	920	919	920	920	925	920.5
p5.2.o	1020	1011	1020	1012	1010	1010	1010	1010
p5.2.p	1150	1143.5	1150	1150	1150	1150	1150	1150
p5.2.q	1195	1194	1195	1192.5	1195	1193	1195	1195
p5.2.r	1260	1258.5	1260	1257.5	1260	1259	1260	1256.5
p5.2.s	1340	1324	1330	1325	1330	1323.5	1330	1324
p5.2.t	1400	1382	1400	1377	1400	1379.5	1400	1382
p5.2.u	1460	1452.5	1460	1447	1460	1457.5	1460	1448
p5.2.v	1505	1491.5	1495	1487	1500	1496.5	1495	1486.5
p5.2.w	1560	1537.5	1555	1541.5	1555	1549.5	1555	1536
p5.2.x	1610	1595.5	1610	1586.5	1610	1607	1610	1593.5
p5.2.y	1645	1631.5	1645	1633.5	1645	1631.5	1645	1632
p5.2.z	1680	1672.5	1680	1680	1680	1673	1680	1677
p5.3.b	15	15	15	15	15	15	15	15
p5.3.c	20	20	20	20	20	20	20	20
p5.3.d	60	60	60	60	60	60	60	60
p5.3.e	95	95	95	95	95	95	95	95
p5.3.f	110	110	110	110	110	110	110	110
p5.3.g	185	185	185	185	185	185	185	185
p5.3.h	260	260	260	260	260	260	260	260
p5.3.i	335	335	335	335	335	335	335	335
p5.3.j	470	470	470	470	470	470	470	470
p5.3.k	495	495	495	495	495	495	495	495
p5.3.l	595	590	595	586	595	584	595	584
p5.3.m	650	649.5	650	649.5	650	649	650	649
p5.3.n	755	755	755	755	755	755	755	755
p5.3.o	870	865	870	864.5	870	867.5	870	864
p5.3.p	990	990	990	990	990	989	990	990
p5.3.q	1070	1061.5	1065	1056.5	1065	1057.5	1065	1056
p5.3.r	1125	1114.5	1120	1113	1125	1114.5	1125	1114.5
p5.3.s	1190	1187	1190	1180.5	1190	1178.5	1185	1179

续表

算例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p5.3.t	1260	1251	1250	1246.5	1255	1246.5	1260	1250.5
p5.3.u	1345	1336	1330	1319	1335	1320	1335	1326
p5.3.v	1425	1402	1425	1412.5	1425	1414.5	1420	1398.5
p5.3.w	1485	1458	1465	1455	1465	1452	1465	1452.5
p5.3.x	1540	1513.5	1535	1523.5	1540	1518	1540	1522
p5.3.y	1590	1555	1590	1552.5	1590	1547.5	1590	1552.5
p5.3.z	1635	1610	1635	1616.5	1635	1623	1635	1615.5
p5.4.c	20	20	20	20	20	20	20	20
p5.4.d	20	20	20	20	20	20	20	20
p5.4.e	20	20	20	20	20	20	20	20
p5.4.f	80	80	80	80	80	80	80	80
p5.4.g	140	140	140	140	140	140	140	140
p5.4.h	140	140	140	140	140	140	140	140
p5.4.i	240	240	240	240	240	240	240	240
p5.4.j	340	340	340	340	340	340	340	340
p5.4.k	340	340	340	340	340	340	340	340
p5.4.l	430	429.5	430	428	430	428	430	428
p5.4.m	555	554	555	552	555	551.5	555	553
p5.4.n	620	620	620	620	620	620	620	620
p5.4.o	690	690	690	689.5	690	690	690	689.5
p5.4.p	765	758	760	755	760	752	760	753
p5.4.q	860	851	860	837.5	860	847	860	839.5
p5.4.r	960	960	960	960	960	958	960	954
p5.4.s	1030	1020	1030	1017	1030	1019.5	1030	1011.5
p5.4.t	1160	1152	1160	1134.5	1160	1139.5	1160	1131
p5.4.u	1300	1300	1300	1274.5	1300	1260	1300	1282.5
p5.4.v	1320	1320	1320	1292.5	1320	1297	1320	1300.5
p5.4.w	1390	1373.5	1380	1374	1390	1374.5	1380	1374.5
p5.4.x	1450	1443	1450	1440.5	1450	1439	1450	1441
p5.4.y	1520	1513	1510	1483	1510	1492	1500	1485
p5.4.z	1620	1585.5	1620	1567	1575	1549	1580	1553.5



表附5　4种构造方法在第6个数据集中的实验结果

算　例	串行法		确定性同步法		随机性同步法		同时法	
	最大值	平均值	最大值	平均值	最大值	平均值	最大值	平均值
p6.2.d	192	189	192	186.6	192	188.4	192	188.4
p6.2.e	360	359.4	360	358.8	360	357	360	360
p6.2.f	588	587.4	588	585.6	588	585	588	586.8
p6.2.g	660	660	660	660	660	660	660	660
p6.2.h	780	780	780	780	780	780	780	780
p6.2.i	888	888	888	888	888	888	888	888
p6.2.j	948	947.4	948	948	948	948	948	948
p6.2.k	1032	1032	1032	1032	1032	1032	1032	1032
p6.2.l	1116	1111.2	1110	1106.4	1116	1111.2	1116	1110.6
p6.2.m	1188	1184.4	1188	1175.4	1188	1182.6	1188	1183.8
p6.2.n	1260	1230.6	1260	1234.8	1254	1230.6	1260	1235.4
p6.3.g	282	278.4	282	277.2	282	276.6	282	277.8
p6.3.h	444	427.8	444	427.8	438	428.4	438	430.2
p6.3.i	642	640.8	642	640.8	642	639.6	642	638.4
p6.3.j	828	825.6	828	825	828	825	828	825.6
p6.3.k	894	888.6	888	888	888	888	894	888.6
p6.3.l	1002	996	1002	996	1002	993	1002	996
p6.3.m	1080	1071.6	1074	1069.8	1080	1071.6	1080	1074
p6.3.n	1170	1159.2	1164	1160.4	1164	1155	1164	1159.8
p6.4.j	366	363	366	362.4	366	361.8	366	363
p6.4.k	528	525	528	522	528	517.8	528	520.2
p6.4.l	696	671.4	696	675.6	696	679.2	696	674.4
p6.4.m	912	885.6	912	873.6	912	876.6	912	886.2
p6.4.n	1068	1061.4	1068	1062	1068	1062	1068	1066.2